# Oracle® Rdb for OpenVMS

# Table of Contents

# Table of Contents

# Table of Contents

# Table of Contents

# Table of Contents

# Table of Contents

# Table of Contents

# Table of Contents

# Table of Contents

# Table of Contents

# Table of Contents

# Oracle® Rdb for OpenVMS

# Release Notes

Release 7.1.2.4

# May 2004

Oracle Rdb Release Notes, Release 7.1.2.4 for OpenVMS

# Contents

# Preface

# Purpose of This Manual

This manual contains release notes for Oracle Rdb Release 7.1.2.4. The notes describe changed and enhanced features; upgrade and compatibility information; new and existing software problems and restrictions; and software and documentation corrections.

# Intended Audience

This manual is intended for use by all Oracle Rdb users. Read this manual before you install, upgrade, or use Oracle Rdb Release 7.1.2.4.

# Document Structure

This manual consists of the following chapters:

| | |
|---|---|
| Chapter 1 | Describes how to install Oracle Rdb Release 7.1.2.4. |
| Chapter 2 | Describes software errors corrected in Oracle Rdb Release 7.1.2.4. |
| Chapter 3 | Describes software errors corrected in Oracle Rdb Release 7.1.2.3. |
| Chapter 4 | Describes software errors corrected in Oracle Rdb Release 7.1.2.2. |
| Chapter 5 | Describes software errors corrected in Oracle Rdb Release 7.1.2.1. |
| Chapter 6 | Describes software errors corrected in Oracle Rdb Release 7.1.2. |
| Chapter 7 | Describes enhancements introduced in Oracle Rdb Release 7.1.2.4. |
| Chapter 8 | Provides information not currently available in the Oracle Rdb documentation set. |
| Chapter 9 | Describes problems, restrictions, and workarounds known to exist in Oracle Rdb Release 7.1.2.4. |

# Chapter 1
# Installing Oracle Rdb Release 7.1.2.4

This software update is installed using the standard OpenVMS Install Utility.

NOTE

***All Oracle Rdb Release 7.1 kits are full kits. There is no requirement to install any prior release of Oracle Rdb when installing new Rdb Release 7.1 kits.***

## 1.1 Alpha EV7 Processor Support

For this release of Oracle Rdb, the Alpha EV7 (also known as the Alpha 21364) processor is the newest processor supported.

# 1.2 Oracle Rdb V7.1 Version Numbering Enhancement

Previously, the Oracle Rdb version number was specified as 4 digits (for example, version "7.1.0.2"). Starting with Oracle Rdb Release 7.1.1, an additional, fifth, digit has been added to the kit version number. This new digit is intended to indicate an optimization level of the Rdb software. The use of this new digit is to indicate a "generic" kit (final digit of zero) for all Alpha processors or a "performance" kit that will run on a subset of the supported platforms (final digit of 1). In the future, additional values may be specified to indicate other performance or platform options.

For Oracle Rdb Release 7.1.2.4, the two kits are 7.1.2.4.0 (compiled for all Alpha processor types) and 7.1.2.4.1 (compiled for EV56 and later Alpha processors). These kits offer identical functionality and differ only in a potential performance difference.

# 1.3 Requirements

The following conditions must be met in order to install this software:

- Oracle Rdb must be shutdown before you install this update kit. That is, the command file SYS$STARTUP:RMONSTOP71.COM should be executed before proceeding with this installation. If you have an OpenVMS cluster, you must shutdown the Rdb 7.1 monitor on all nodes in the cluster before proceeding.
- The installation requires approximately 280,000 blocks for OpenVMS Alpha systems.
- If you are running Hot Standby and you are upgrading from a version of Oracle Rdb 7.1 prior to 7.1.1, you must install this kit on both the master and the standby systems prior to restarting Hot Standby. This requirement is necessary due to changes to the message format used to transmit journal state information from the master to the standby system.

# 1.4 Invoking VMSINSTAL

To start the installation procedure, invoke the VMSINSTAL command procedure as in the following examples.

To install the Oracle Rdb for OpenVMS Alpha kit that is compiled to run on all Alpha platforms:

```
@SYS$UPDATE:VMSINSTAL RDBV71240AM device−name OPTIONS N
```

To install the Oracle Rdb for OpenVMS Alpha kit that is performance targeted for Alpha EV56 and later platforms:

```
@SYS$UPDATE:VMSINSTAL RDBV71241AM device−name OPTIONS N
```

***device−name***

Use the name of the device on which the media is mounted. If the device is a disk drive, you also need to specify a directory. For example: *DKA400:[RDB.KIT]*

***OPTIONS N***

This parameter prints the release notes.

The full Oracle Rdb Release 7.1 Installation Guide is also available on MetaLink in Adobe Acrobat PDF format:

```
Top Tech Docs\Oracle Rdb\Documentation\Rdb 7.1 Installation and Configuration
Guide
```

# 1.5 Stopping the Installation

To stop the installation procedure at any time, press Ctrl/Y. When you press Ctrl/Y, the installation procedure deletes all files it has created up to that point and exits. You can then start the installation again.

If VMSINSTAL detects any problems during the installation, it notifies you and a prompt asks if you want to continue. You might want to continue the installation to see if any additional problems occur. However, the copy of Oracle Rdb installed will probably not be usable.

# 1.6 After Installing Oracle Rdb

This update provides a new Oracle Rdb Oracle TRACE facility definition. Any Oracle TRACE selections that reference Oracle Rdb will need to be redefined to reflect the new facility version number for the updated Oracle Rdb facility definition, "RDBVMSV7.1−2".

If you have Oracle TRACE installed on your system and you would like to collect for Oracle Rdb, you must insert the new Oracle Rdb facility definition included with this update kit.

The installation procedure inserts the Oracle Rdb facility definition into a library file called EPC$FACILITY.TLB. To be able to collect Oracle Rdb event−data using Oracle TRACE, you must move this facility definition into the Oracle TRACE administration database. Perform the following steps:

1. Extract the definition from the facility library to a file (in this case, RDBVMS.EPC$DEF).

   ```
   $ LIBRARY /TEXT /EXTRACT=RDBVMSV7.1−2 −
   _$ /OUT=RDBVMS.EPC$DEF SYS$SHARE:EPC$FACILITY.TLB
   ```
2. Insert the facility definition into the Oracle TRACE administration database.

   ```
   $ COLLECT INSERT DEFINITION RDBVMS.EPC$DEF /REPLACE
   ```

Note that the process executing the INSERT DEFINITION command must use the version of Oracle Rdb that matches the version used to create the Oracle TRACE administration database or the INSERT DEFINITION command will fail.

# 1.7 Patches for OpenVMS V7.3−1

Several problems that affect installations using Oracle Rdb on OpenVMS V7.3−1 are corrected in patch kits available from HP OpenVMS support. Oracle recommends that you consult with Hewlett−Packard and install these patch kits (or their replacements) to correct or avoid the following problems:

- VMS731_SYS−V0400 corrects the following problems seen with Oracle Rdb:
    - When using Oracle Rdb Galaxy support, or memory−resident global sections, processes enter a permanent RWAST state at image exit. The system must be rebooted to remove the process and continue normal operations. Note that when using Oracle Rdb Release 7.1.2 databases with SHARED MEMORY IS PROCESS RESIDENT attribute, the Row Cache feature and caches with the SHARED MEMORY IS SYSTEM, LARGE MEMORY IS ENABLED, or RESIDENT attributes, or in an OpenVMS Galaxy configuration with Oracle Rdb Galaxy support enabled, you are at an elevated risk of experiencing this problem.
      Configurations that do not have this patch, or it's future replacement, applied will not be supported by Oracle if the SHARED MEMORY IS PROCESS RESIDENT, the Row Cache, or Galaxy support features are in use. If you are not using these features, then the patch or it's replacement is not mandatory. However, Oracle still strongly recommends that it be used.
    - Applications using the Oracle Rdb Row Cache or AIJ Log Server (ALS) features would sometimes have their server processes hang in HIB (hibernate) state.
- VMS731_SYSLOA−V0100 corrects the following problem seen with Oracle Rdb:
    - In an OpenVMS cluster environment, unreported deadlocks and hangs can occur. This problem is sometimes characterized by an Oracle Rdb blocking lock incorrectly being shown as owned by the system (in other words, with a zero PID).

# 1.8 Oracle Rdb Release 7.1.2.4.1 Optimized for Alpha EV56 (21164A Processor Chip) and Later Platforms

Oracle will be releasing Oracle Rdb 7.1 and later kits in parallel build streams − a "generic" kit that will run on all certified and supported Alpha platforms as well as a "performance" kit that will run on a subset of the supported platforms. The performance kit is intended for those customers with "newer" Alpha processor chips who need higher levels of performance than are offered by the generic kits. The performance kits are otherwise functionally identical to the generic kits.

Oracle will continue to release both types of kits for Oracle Rdb Release 7.1 as long as there is significant customer interest in the generic kit.

For improved performance on current generation Alpha processors, Oracle Rdb Release 7.1.2.4.1 is compiled explicitly for Alpha EV56 and later systems. This version of Oracle Rdb requires a system with a minimum Alpha processor chip of EV56 and a maximum processor chip of Alpha EV7 (known as the Alpha 21364).

Oracle Rdb Release 7.1.2.4.1 is functionally equivalent to Oracle Rdb Release 7.1.2.4.0 and was built from the same source code. The only difference is a potentially improved level of performance. Oracle Rdb Releases 7.1.2.4.0 and 7.1.2.4.1 are certified on all supported Alpha processor types (up to and including the Alpha EV7 processor).

In Release 7.1.2.4.1, Oracle Rdb is explicitly compiled for EV56 and later Alpha processors such that the generated instruction stream can utilize the byte/word extension (BWX) of the Alpha architecture. Additionally, this kit is compiled with instruction tuning biased for performance of Alpha EV6 and later systems that support quad−issue instruction scheduling.

Note that you should not install Release 7.1.2.4.1 of Oracle Rdb on Alpha EV4, EV45 or EV5 systems. These processor types do not support the required byte/word extension (BWX) of the Alpha architecture. Also ensure that all systems in a cluster sharing the system disk are using a minimum of the Alpha EV56 processor.

To easily determine the processor type of a running OpenVMS Alpha system, use the CLUE CONFIG command of the OpenVMS System Dump Analyzer utility (accessed with the ANALYZE/SYSTEM command). The "CPU TYPE" field indicates the processor type as demonstrated in the following example from an HP AlphaServer GS140 6/525 system with an EV6 (21264) processor:

```
$ ANALYZE/SYSTEM
SDA> CLUE CONFIG
System Configuration:
    .
    .
    .
Per−CPU Slot Processor Information:
CPU ID    00                  CPU State    rc,pa,pp,cv,pv,pmv,pl
CPU Type  EV6  Pass 2.3 (21264)
PAL Code  1.96−1              Halt PC      00000000.20000000
    .
    .
    .
```

## 1.8.1 AlphaServer 4000 EV56 299Mhz Not Supported by Oracle Rdb Release Optimized for Alpha EV56 Processor

Oracle Rdb releases that are optimized for the Alpha EV56 and later processors are not able to run on the AlphaServer 4000 with the 299Mhz EV56 processor. Though this CPU claims to be an EV56, it does not, in fact, implement the byte/word instruction set as required.

According to information on the HP web site, this problem may be present in the AlphaServer 4000 or 4100 systems with a processor module of KN304−FA or KN304−FB. The systems effected appear to include the AlphaServer 4x00 5/300 pedestal, cabinet and rackmount systems: DA−51FAB−ED/−FD/−GB or DA−53GEB−CA/−EA/−FA/−GA.

The indicated CPU is not able to run Oracle Rdb releases that are optimized for the Alpha EV56 and later processors. This effects Oracle Rdb Releases optimized for the Alpha EV56 and later processors.

Possible workarounds include updating the system to an EV56 module for the AlphaServer 4x00 that is later than the KN304−FA or FB (ie a clock speed greater than 300Mhz). Some of the possible modules would include: KN304−AA 400mhz, KN304−DA 466mhz, B3005−CA 533mhz, B3006−EB 600mhz.

Otherwise, an Oracle Rdb release that is not optimized for the Alpha EV56 and later processors must be used (such as Oracle Rdb Releases 7.1.2.4.0)

Please contact your HP AlphaServer hardware vendor for additional information.

# 1.9 Maximum OpenVMS Version Check Added

As of Oracle Rdb7 Release 7.0.1.5, a maximum OpenVMS version check has been added to the product. Oracle Rdb has always had a minimum OpenVMS version requirement. With 7.0.1.5 and for all future Oracle Rdb releases, we have expanded this concept to include a maximum VMS version check and a maximum supported processor hardware check. The reason for this check is to improve product quality.

OpenVMS Version 7.3−x is the maximum supported version of OpenVMS.

The check for the OpenVMS operating system version and supported hardware platforms is performed both at installation time and at runtime. If either a non−certified version of OpenVMS or hardware platform is detected during installation, the installation will abort. If a non−certified version of OpenVMS or hardware platform is detected at runtime, Oracle Rdb will not start.

## 1.10 VMS$MEM_RESIDENT_USER Rights Identifier Required

Oracle Rdb Version 7.1 introduced additional privilege enforcement for the database or row cache attributes RESIDENT, SHARED MEMORY IS SYSTEM and LARGE MEMORY IS ENABLED. If a database utilizes any of these features, then the user account that opens the database must be granted the VMS$MEM_RESIDENT_USER rights identifier.

Oracle recommends that the RMU/OPEN command be used when utilizing these features.

# Chapter 2
# Software Errors Fixed in Oracle Rdb Release 7.1.2.4

This chapter describes software errors that are fixed by Oracle Rdb Release 7.1.2.4.

# 2.1 Software Errors Fixed That Apply to All Interfaces

## 2.1.1 RDBSERVER71.EXE Installed As Known Image

The Oracle Rdb startup procedure (RMONSTART71.COM) now installs the RDBSERVER71.EXE image file as a known image.

## 2.1.2 MINUS Query With GROUP BY Clauses Returns Wrong Results

Bug 2882908

The following MINUS query with GROUP BY clauses should return 0 rows.

```
set flags 'strategy,detail';
select employee_id, postal_code from employees
     where employee_id >= '00415'
    group by employee_id, postal_code
minus
select employee_id, postal_code from employees
     where employee_id >= '00415'
    group by employee_id, postal_code
;
Tables:
  0 = EMPLOYEES
  1 = EMPLOYEES
Merge of 1 entries
  Merge block entry 1
  Cross block of 2 entries
    Cross block entry 1
      Reduce: 0.EMPLOYEE_ID, 0.POSTAL_CODE
      Sort: 0.EMPLOYEE_ID(a), 0.POSTAL_CODE(a)
      Leaf#01 BgrOnly 0:EMPLOYEES Card=100
        Bool: 0.EMPLOYEE_ID >= '00415'
        BgrNdx1 EMP_EMPLOYEE_ID [1:0] Fan=17
          Keys: 0.EMPLOYEE_ID >= '00415'
    Cross block entry 2
      Conjunct: <agg0> = 0
      Aggregate-F1: 0:COUNT-ANY (<subselect>)
      Reduce: 1.EMPLOYEE_ID, 1.POSTAL_CODE
      Sort: 1.EMPLOYEE_ID(a), 1.POSTAL_CODE(a)
      Conjunct: (0.POSTAL_CODE = 1.POSTAL_CODE) OR (MISSING (0.POSTAL_CODE) AND
                MISSING (1.POSTAL_CODE))
      Conjunct: 1.EMPLOYEE_ID >= '00415'
      OR index retrieval
        Conjunct: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
        Get     Retrieval by index of relation 1:EMPLOYEES
          Index name  EMP_EMPLOYEE_ID [1:1]      Direct lookup
            Keys: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
        Conjunct: NOT (0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND MISSING (0.EMPLOYEE_ID
                  ) AND MISSING (1.EMPLOYEE_ID)
        Get     Retrieval by index of relation 1:EMPLOYEES
          Index name  EMP_EMPLOYEE_ID [1:1]      Direct lookup
            Keys: MISSING (1.EMPLOYEE_ID)
 EMPLOYEE_ID   POSTAL_CODE
```

```
 00415         03442
 00416         03809
 00418         03602
 00435         03455
 00471         03301
5 rows selected
```

The MINUS operator is a new SQL feature introduced in Oracle Rdb 7.1 and this query applies the MINUS operator between two SELECT statements with GROUP BY clauses.

The only workaround for this query is to add the clause OPTIMIZE FOR SEQUENTIAL ACCESS at the end of the query, as in the following example.

```
select employee_id, postal_code from employees
     where employee_id >= '00415'
    group by employee_id, postal_code
minus
select employee_id, postal_code from employees
     where employee_id >= '00415'
    group by employee_id, postal_code
optimize for sequential access
;
Tables:
  0 = EMPLOYEES
  1 = EMPLOYEES
Merge of 1 entries
  Merge block entry 1
  Cross block of 2 entries
    Cross block entry 1
      Reduce: 0.EMPLOYEE_ID, 0.POSTAL_CODE
      Sort: 0.EMPLOYEE_ID(a), 0.POSTAL_CODE(a)
      Conjunct: 0.EMPLOYEE_ID >= '00415'
      Get     Retrieval sequentially of relation 0:EMPLOYEES
    Cross block entry 2
      Conjunct: <agg0> = 0
      Aggregate-F1: 0:COUNT-ANY (<subselect>)
      Reduce: 1.EMPLOYEE_ID, 1.POSTAL_CODE
      Sort: 1.EMPLOYEE_ID(a), 1.POSTAL_CODE(a)
      Conjunct: ((0.EMPLOYEE_ID = 1.EMPLOYEE_ID) OR (MISSING (0.EMPLOYEE_ID)
                 AND MISSING (1.EMPLOYEE_ID))) AND ((0.POSTAL_CODE =
                1.POSTAL_CODE) OR (MISSING (0.POSTAL_CODE) AND MISSING (
                1.POSTAL_CODE)))
      Conjunct: 1.EMPLOYEE_ID >= '00415'
      Get     Retrieval sequentially of relation 1:EMPLOYEES
0 rows selected
```

The query also works if both GROUP BY clauses are removed, as in the following example.

```
select employee_id, postal_code from employees
     where employee_id >= '00415'
minus
select employee_id, postal_code from employees
     where employee_id >= '00415'
;
Tables:
  0 = EMPLOYEES
  1 = EMPLOYEES
Reduce: 0.EMPLOYEE_ID, 0.POSTAL_CODE
Sort: 0.EMPLOYEE_ID(a), 0.POSTAL_CODE(a)
```

```
Merge of 1 entries
  Merge block entry 1
  Cross block of 2 entries
    Cross block entry 1
      Leaf#01 BgrOnly 0:EMPLOYEES Card=100
        Bool: 0.EMPLOYEE_ID >= '00415'
        BgrNdx1 EMP_EMPLOYEE_ID [1:0] Fan=17
          Keys: 0.EMPLOYEE_ID >= '00415'
    Cross block entry 2
      Conjunct: <agg0> = 0
      Aggregate-F1: 0:COUNT-ANY (<subselect>)
      Conjunct: ((0.EMPLOYEE_ID = 1.EMPLOYEE_ID) OR (MISSING (0.EMPLOYEE_ID)
                  AND MISSING (1.EMPLOYEE_ID))) AND ((0.POSTAL_CODE =
                 1.POSTAL_CODE) OR (MISSING (0.POSTAL_CODE) AND MISSING (
                 1.POSTAL_CODE))) AND (1.EMPLOYEE_ID >= '00415')
      OR index retrieval
        Get     Retrieval by index of relation 1:EMPLOYEES
          Index name  EMP_EMPLOYEE_ID [1:1]      Direct lookup
            Keys: 0.EMPLOYEE_ID = 1.EMPLOYEE_ID
        Conjunct: NOT (0.EMPLOYEE_ID = 1.EMPLOYEE_ID) AND MISSING (0.EMPLOYEE_ID
                   ) AND MISSING (1.EMPLOYEE_ID)
        Get     Retrieval by index of relation 1:EMPLOYEES
          Index name  EMP_EMPLOYEE_ID [1:1]      Direct lookup
            Keys: MISSING (1.EMPLOYEE_ID)
0 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.1.3 Bugchecks in RDMS$$KOD_EVAL_BOOL When Using Bitmap Scan

Bug 3430322

When using the bitmap scan feature, it was possible that bugcheck dumps could occur with an exception at *RDMS$$KOD_EVAL_BOOL + 000000BC*.

In the following example, the standard MF_PERSONNEL database has been modified with additional indices.

```
SQL> set flags 'bitmap'
SQL> select * from employees
cont>         where employee_id>'003'
cont>         and (sex='M' or last_name<'B')
cont>         and first_name>'T';
%RDMS-I-BUGCHKDMP, generating bugcheck dump file USR:[DIR]RDSBUGCHK.DMP;
```

The problem occurred because the bitmap scan feature used an "OR" index list for the indexes on the sex and last_name fields. In addition, the first "OR" index was a candidate for index estimation and the second "OR" index could not be estimated because it was partitioned. If the first background index in the "OR" list was not a candidate for estimation by the dynamic optimizer, for example because it was a partitioned index, then none of the OR indexes would be estimated. In this situation, no bugcheck would result and the query would execute successfully.

If the first background index in the "OR" list was a candidate for estimation, the dynamic optimizer would attempt estimation on all indexes in the "OR" list even if they were partitioned. In this situation, the bugcheck described above would result.

Oracle Rdb now correctly detects which indices can be estimated, and for those that cannot, the static optimizer's estimated cost is used.

The estimation execution summary line will only show estimation as not done for the "OR" index list if none of the "OR" indexes were estimated. The *SET FLAGS 'EXECUTION,DETAIL'* command can be used to show which indexes in the "OR" index list were actually estimated.

This problem can be avoided if the first index in the "OR" list cannot be estimated, for example if the first index is a partitioned index. The problem can also be avoided by ensuring that all indexes in the "OR" list can be estimated.

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.1.4 Database Corruption When LOCKING IS PAGE LEVEL Enabled

Storage areas created using the "LOCKING IS PAGE LEVEL" option would sometimes have missing updates. This problem would occur when there was a lot of contention for pages in the storage area.

For example, if an index update were lost, IDXDATMIS errors may have been displayed by RMU/VERIFY:

```
%RMU-W-IDXDATMIS, Index ALL_DEPT does not point to a row in table
DEPARTMENT_REC.
      Logical dbkey of the missing row is 149:302:5.
```

This problem can be avoided by using the "LOCKING IS ROW LEVEL" option, which is the default.

Note that the "LOCKING IS PAGE LEVEL" option works best with applications that have little contention between users for database pages. Also, ideally, transactions should be short in duration and most if not all pages used by the transaction should be able to simultaneously reside in the buffers allocated to the process.

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.1.5 Wrong Results With Superfluous ORDER BY Clause

Bug 3128602

A simple COUNT query returned a single row, the count. The same query with an additional (and unnecessary) ORDER BY clause, returned the wrong count. For the examples in this section, the test database is the Rdb PERSONNEL database. An additional index was added to the SALARY_HISTORY table, as follows:

```
create index test_index on salary_history (salary_start, employee_id);
```

Below is the original query returning the correct count.

```
select count (*) from salary_history sh, employees e
  where sh.employee_id = e.employee_id;
```

```
         729
1 row selected
```

Below is the same query with the additional ORDER BY clause returning the incorrect count.

```
select count (*) from salary_history sh, employees e
  where sh.employee_id = e.employee_id
  order by sh.salary_start, sh.employee_id;

           0
1 row selected
```

As a workaround, remove the ORDER BY clause. It is not necessary to sort an answer that contains only one row.

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.1.6 Table Constraint Should Fail on Updating a Row

Bug 3434648

The customer updates a row in a table, changing the value in one column from NULL to some text string. Under Oracle Rdb 7.0.6.3, the report claims, a constraint on that table correctly reports that the update is in error. However, in Oracle Rdb Release 7.0.7.1 the constraint failure does not occur.

A simple reproducer using a SELECT statement is created from the UPDATE/COMMIT statement of the original BUG report where the CHECK clause defined in the table constraint contains four OR predicates, as in the following example.

```
predicate−1 OR predicate−2 OR predicate−3 OR predicate−4
```

```
which is inverted using the NOT operator, such as:
```

```
NOT (predicate−1 OR predicate−2 OR predicate−3 OR predicate−4)
```

The query selects "Constraint violation" on the ALIAS_LOC table where the dbkey is the target row that UPDATE performs on. It should find the row that gets the constraint violation when the conditions in the predicates are not met.

```
set flags 'strategy,detail';

declare :dbk bigint;
select dbkey into :dbk from ALIAS_LOC A limit to 1 row;

select 'Constraint Failure' from ALIAS_LOC A
  where a.dbkey = :dbk and
    NOT (
        ! Part 1
         (A.LOCATION_1_NAME is null and
          A.LOCATION_2_NAME is null and
          A.LOCATION_3_NAME is null)
        OR
        ! Part 2
         ((A.LOCATION_2_NAME is null and        ! <== missing in Cross block 3
           A.LOCATION_3_NAME is null)           ! <== of the strategy
```

```
          and exists
          (select * from LOCATION C10
           where
             C10.ORGANIZATION_NAME = A.ORGANIZATION_NAME and
             C10.UNIT_NAME         = A.UNIT_NAME         and
             C10.PARENT_LOCATION_NAME is null            and
             C10.LOCATION_NAME     = A.LOCATION_1_NAME
          )             ! end of select
         )              ! end of or
        OR
        ! Part 3
        ((A.LOCATION_3_NAME is null)
         and exists
         (select * from LOCATION C11, LOCATION C12
           where
               C11.ORGANIZATION_NAME    = A.ORGANIZATION_NAME and
               C11.UNIT_NAME            = A.UNIT_NAME         and
               C11.PARENT_LOCATION_NAME is null               and
               C11.LOCATION_NAME        = A.LOCATION_1_NAME   and
               C12.PARENT_LOCATION_NAME = C11.LOCATION_NAME   and
               C12.LOCATION_NAME        = A.LOCATION_2_NAME
          )             ! end of select
         )              ! end of or
        OR
        ! Part 4
        (exists
         (select * from LOCATION C13, LOCATION C14, LOCATION C15
          where
               C13.ORGANIZATION_NAME    = A.ORGANIZATION_NAME and
               C13.UNIT_NAME            = A.UNIT_NAME         and
               C13.PARENT_LOCATION_NAME is null               and
               C13.LOCATION_NAME        = A.LOCATION_1_NAME   and
               C14.PARENT_LOCATION_NAME = C13.LOCATION_NAME   and
               C14.LOCATION_NAME        = A.LOCATION_2_NAME   and
               C15.PARENT_LOCATION_NAME = C14.LOCATION_NAME   and
               C15.LOCATION_NAME        = A.LOCATION_3_NAME
         )              ! end of select
        )               ! end of or
      )
  limit to 1 row;
Tables:
  0 = ALIAS_LOC
  1 = LOCATION
  2 = LOCATION
  3 = LOCATION
  4 = LOCATION
  5 = LOCATION
  6 = LOCATION
Firstn: 1
Cross block of 4 entries
  Cross block entry 1
    Conjunct: NOT MISSING (0.LOC_NAME_1) OR NOT MISSING (0.LOC_NAME_2) OR NOT
              MISSING (0.LOC_NAME_3)
    Conjunct: 0.DBKEY = <var0>
    Firstn: 1
    Get     Retrieval by DBK of relation 0:ALIAS_LOC
  Cross block entry 2
    Conjunct: NOT MISSING (0.LOC_NAME_3) OR (<agg0> = 0)
    Aggregate-F1: 0:COUNT-ANY (<subselect>)
    Cross block of 2 entries
      Cross block entry 1
```

```
        Index only retrieval of relation 2:LOCATION
          Index name  LOCATION_NDX [4:4]         Direct lookup
            Keys: (2.ORG_NAME = 0.ORG_NAME) AND (2.UNIT_NAME = 0.UNIT_NAME) AND
                  (MISSING (2.P_LOC_NAME)) AND (2.LOC_NAME = 0.LOC_NAME_1)
      Cross block entry 2
        Conjunct: (3.P_LOC_NAME = 2.LOC_NAME) AND (3.LOC_NAME = 0.LOC_NAME_2)
        Index only retrieval of relation 3:LOCATION
          Index name  LOCATION_NDX [0:0]
  Cross block entry 3
    Conjunct: <agg1> = 0        ! <=== See Note below.
    Aggregate-F1: 1:COUNT-ANY (<subselect>)
    Index only retrieval of relation 1:LOCATION
      Index name  LOCATION_NDX [4:4]         Direct lookup
        Keys: (1.ORG_NAME = 0.ORG_NAME) AND (1.UNIT_NAME = 0.UNIT_NAME) AND (
              MISSING (1.P_LOC_NAME)) AND (1.LOC_NAME = 0.LOC_NAME_1)
  Cross block entry 4
    Conjunct: <agg2> = 0
    Aggregate-F1: 2:COUNT-ANY (<subselect>)
    Cross block of 3 entries
      Cross block entry 1
        Index only retrieval of relation 4:LOCATION
          Index name  LOCATION_NDX [4:4]         Direct lookup
            Keys: (4.ORG_NAME = 0.ORG_NAME) AND (4.UNIT_NAME = 0.UNIT_NAME) AND
                  (MISSING (4.P_LOC_NAME)) AND (4.LOC_NAME = 0.LOC_NAME_1)
      Cross block entry 2
        Conjunct: (5.P_LOC_NAME = 4.LOC_NAME) AND (5.LOC_NAME = 0.LOC_NAME_2)
        Index only retrieval of relation 5:LOCATION
          Index name  LOCATION_NDX [0:0]
      Cross block entry 3
        Conjunct: (6.P_LOC_NAME = 5.LOC_NAME) AND (6.LOC_NAME = 0.LOC_NAME_3)
        Index only retrieval of relation 6:LOCATION
          Index name  LOCATION_NDX [0:0]
0 rows selected
```

NOTE:: Note that the following conjunct under the Cross block entry 3 should contain the following predicates, but that is missing:

```
    Conjunct: NOT MISSING (0.LOC_NAME_2) OR NOT MISSING (0.LOC_NAME_3) OR
            (<agg1> = 0)
```

This problem was caused by the fix for Bug 2285818 where the query contains predicates shared by another part of the OR expression tree.

This query also contains the following similar predicates shared by another part of the OR expression tree.

```
    NOT (
        ! Part 1
         (A.LOCATION_1_NAME is null and
          A.LOCATION_2_NAME is null and
          A.LOCATION_3_NAME is null)
        OR
        ! Part 2
         ((A.LOCATION_2_NAME is null and        ! <== shared in PART 1
           A.LOCATION_3_NAME is null)           ! <== shared in PART 1
          and
          ...etc...)
         )              ! end of or
        OR
        ! Part 3
```

```
        ((A.LOCATION_3_NAME is null) and        ! <== shared in PART 1 and 2
         ...etc...)
        OR
        ! Part 4
         (...etc...)
        );
```

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.1.7 Zigzag Match Query with Descending Index Segment Returns Wrong Results

Bug 3514413

The following zigzag match query with a descending index segment should select 6 rows, but selects 1 row instead.

```
set flags 'strategy,detail';
SELECT   MBR_ID, ACCNT_TYPE, ACCNT_NO, CNTR_PRCE, TRB.SERI_CODE
    FROM   TRD_BOOK_TB TRB, SERIES_DAILY_TB SED, SERIES_TB SER
    WHERE   TRB.DRV_MKT_ID = 'BF'
      AND   TRB.ULY_ID = '61'
      AND   SED.YYMMDD    =   '20040309'
      AND   TRB.SERI_CODE =   SED.SERI_CODE
      AND   SED.SERI_CODE =   SER.SERI_CODE
      AND   SER.ACT_DATE  <=  '20040309'
      AND   SER.DEL_DATE  >=  '20040309'
;
Tables:
  0 = TRD_BOOK_TB
  1 = SERIES_DAILY_TB
  2 = SERIES_TB
Conjunct: 1.SERI_CODE = 2.SERI_CODE
Match
  Outer loop
    Conjunct: 0.SERI_CODE = 1.SERI_CODE
    Match
      Outer loop
        Sort: 0.SERI_CODE(a)
        Conjunct: (0.DRV_MKT_ID = 'BF') AND (0.ULY_ID = '61')
        Get     Retrieval sequentially of relation 0:TRD_BOOK_TB
      Inner loop      (zig-zag)
        Index only retrieval of relation 1:SERIES_DAILY_TB
          Index name  SERIES_DAILY_IDX [1:1]
            Keys: 1.YYMMDD = '20040309'
  Inner loop
    Temporary relation
    Sort: 2.SERI_CODE(a)
    Conjunct: 2.ACT_DATE <= '20040309'
    Leaf#01 BgrOnly 2:SERIES_TB Card=1002
      Bool: 2.DEL_DATE >= '20040309'
      BgrNdx1 SERIES_IDX [0:1] Fan=11
        Keys: 2.ACT_DATE <= '20040309'
        Bool: 2.DEL_DATE >= '20040309'
 TRB.MBR_ID   TRB.ACCNT_TYPE   TRB.ACCNT_NO    TRB.CNTR_PRCE   TRB.SERI_CODE
```

```
 016          3                 001000027              106.70   KR4161460000
1 row selected
```

As a workaround, the query works if the zigzag match strategy is disabled, as in the following example.

```
SQL> set flags 'nozigzag_match';
  OR
$DEFINE RDMS$$DISABLE_ZIGZAG_MATCH 2

Tables:
  0 = TRD_BOOK_TB
  1 = SERIES_DAILY_TB
  2 = SERIES_TB
Conjunct: 1.SERI_CODE = 2.SERI_CODE
Match
  Outer loop
    Conjunct: 0.SERI_CODE = 1.SERI_CODE
    Match
      Outer loop
        Sort: 0.SERI_CODE(a)
        Conjunct: (0.DRV_MKT_ID = 'BF') AND (0.ULY_ID = '61')
        Get     Retrieval sequentially of relation 0:TRD_BOOK_TB
      Inner loop
        Index only retrieval of relation 1:SERIES_DAILY_TB
          Index name  SERIES_DAILY_IDX [1:1]
            Keys: 1.YYMMDD = '20040309'
  Inner loop
    Temporary relation
    Sort: 2.SERI_CODE(a)
    Conjunct: 2.ACT_DATE <= '20040309'
    Leaf#01 BgrOnly 2:SERIES_TB Card=1002
      Bool: 2.DEL_DATE >= '20040309'
      BgrNdx1 SERIES_IDX [0:1] Fan=11
        Keys: 2.ACT_DATE <= '20040309'
        Bool: 2.DEL_DATE >= '20040309'
 TRB.MBR_ID   TRB.ACCNT_TYPE   TRB.ACCNT_NO    TRB.CNTR_PRCE   TRB.SERI_CODE
 029          3                 001000025              103.90   KR4161430000
 017          0                 001000004              103.90   KR4161430000
 029          1                 001000023              106.45   KR4161460000
 016          3                 001000027              106.45   KR4161460000
 029          2                 001000099              106.70   KR4161460000
 016          3                 001000027              106.70   KR4161460000
6 rows selected
```

The query also works with the zigzag match strategy if the index column YYMMDD of SERIES_DAILY_IDX index is redefined as ASCENDING instead of DESCENDING.

```
drop index SERIES_DAILY_IDX;
create unique index SERIES_DAILY_IDX
    on SERIES_DAILY_TB (
!   YYMMDD  desc,
    YYMMDD  asc,
    SERI_CODE asc);

Tables:
  0 = TRD_BOOK_TB
  1 = SERIES_DAILY_TB
  2 = SERIES_TB
Conjunct: 1.SERI_CODE = 2.SERI_CODE
Match
```

```
  Outer loop
    Conjunct: 0.SERI_CODE = 1.SERI_CODE
    Match
      Outer loop
        Sort: 0.SERI_CODE(a)
        Conjunct: (0.DRV_MKT_ID = 'BF') AND (0.ULY_ID = '61')
        Get     Retrieval sequentially of relation 0:TRD_BOOK_TB
      Inner loop      (zig-zag)
        Index only retrieval of relation 1:SERIES_DAILY_TB
          Index name  SERIES_DAILY_IDX [1:1]
            Keys: 1.YYMMDD = '20040309'
  Inner loop
    Temporary relation
    Sort: 2.SERI_CODE(a)
    Conjunct: 2.ACT_DATE <= '20040309'
    Leaf#01 BgrOnly 2:SERIES_TB Card=1002
      Bool: 2.DEL_DATE >= '20040309'
      BgrNdx1 SERIES_IDX [0:1] Fan=11
        Keys: 2.ACT_DATE <= '20040309'
        Bool: 2.DEL_DATE >= '20040309'
 TRB.MBR_ID   TRB.ACCNT_TYPE   TRB.ACCNT_NO    TRB.CNTR_PRCE   TRB.SERI_CODE
 029          3                001000025              103.90   KR4161430000
 017          0                001000004              103.90   KR4161430000
 029          1                001000023              106.45   KR4161460000
 016          3                001000027              106.45   KR4161460000
 029          2                001000099              106.70   KR4161460000
 016          3                001000027              106.70   KR4161460000
6 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.1.8 New AIJ Full Status

Bugs 613934 and 2101705

When using several circular AIJs, it was possible that all AIJ files would fill up. Then, since updates could not be written anymore to the AIJ files, the database would freeze for the time specified by the time out parameter (default 60 minutes) before shutting down.

In previous releases of Rdb, when all circular AIJs were full, they were marked inaccessible. Therefore, if a process failed, the DBR would not do the redo phase when fast commit was enabled since AIJs were marked inaccessible. In such conditions, it was possible that the failing process would not be recovered properly and would miss its committed updates since its last checkpoint. Since the DBR could not do the redo phase, it would terminate with failure and the database would be shutdown.

At that point, the only way to recover the database properly and ensure that no comitted updates were lost was to restore the database and recover it from the AIJ files.

Now, Rdb will detect all circular AIJ full conditions and will mark them full. The meaning of this new status is:

- It is not possible to write to the AIJ files so journaling is being shutdown.
- It is possible to read AIJ files.

Since Rdb knows that the AIJs are readable, if a process fails then the DBR will perform the redo phase, writing back to the database all committed updates since the last checkpoint.

When journaling is being shutdown on AIJ full, the database will be frozen for the duration of the 'shutdown' parameter. You can identify the AIJ full condition by doing an RMU/DUMP/HEAD=JOURNAL <db_root> which will show:

```
$ RMU/DUMP/HEAD=JOURNAL <db_root>
        :
      - All journals are accessible
          Journaling has been shut-down
          (files are full)
        :
```

While the database is frozen, you can resolve the situation by using one of the following online commands.

1. Add new journal files if you have some empty AIJ slots left. For example:

   ```
   $ RMU/SET AFTER -
   /ADD=(NAME=FOO_AIJ3,FILE=TDB:FOO_AIJ3.AIJ) -
   /ADD=(NAME=FOO_AIJ4,FILE=TDB:FOO_AIJ4.AIJ) db
   ```
2. Do an online, no quiet point after image backup. For example:

   ```
   $ RMU/BACKUP/AFTER/NOQUIET db foo_aijbck.aij
   ```

The backup must be noquietpoint since there is an active transaction waiting to write to the AIJ and so the quiet point lock can not be acquired leaving the backup process hanging waiting for the quiet point lock.

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.1.9 COSI−F−INVCLADTY During Timestamp Conversion

Bug 3544859

When attempting a data type conversion involving a timestamp data type, a COSI−F−INVCLADTY error could occur.

The following example shows this.

```
select * from timetable where departure_time > timestamp
    '2004-02-02 00:00:00.00'

Caused:
%RDB-E-CONVERT_ERROR, invalid or unsupported data conversion
-COSI-F-INVCLADTY, invalid class data type combination in descriptor

Where:
departure_time is TIMESTAMP(2)
```

A possible workaround for the above example might be the following.

```
select * from timetable where cast(departure_time as date vms) >
    date vms ' 2-FEB-2004 00:00:00.00'
```

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.1.10 Wrong Order on Descending and Ascending Sort on Same Column

Bug 3197004

When a query contains ORDER BY and GROUP BY clauses, the Rdb Optimizer attempts to eliminate sorts where possible. One way it does so is by rearranging the order of keys in a GROUP BY clause to match the order in some ORDER BY clause and then it only performs a single sort instead of two sorts.

The following sample query contains multiple GROUP BY and ORDER BY clauses.

```
select  t1.name, t2.name, t1.datum, t2.datum
   from (select name, datum from a
           group by name, datum) t1
        join
        (select name, datum from b
           group by name, datum) t2
        on (t1.name = t2.name and t1.datum = t2.datum)
    group by t1.name, t2.name, t1.datum, t2.datum
    order by t1.name desc, t2.name asc, t1.datum desc, t2.datum asc;
```

Note that the final ORDER BY clause sorts first on the name column in descending order and then on the name column in ascending order. Before the problem was corrected, the Rdb Optimizer incorrectly produced a query strategy that sorted everything in ascending order.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.1.11 Bugcheck at RDMS$FIND_CJOIN + 0000B69C

Bug 3395635

A query worked without incident by itself. However, after a query outline was created and the query was re−executed, a bugcheck in Rdb was reported during query compilation. The bugcheck dump file showed that an access violation had occurred at location RDMS$$FIND_CJOIN + 0000B69C.

The problem was reproducible using the Rdb sample database MF_PERSONNEL. Below are the commands entered to show the problem.

```
create index emp_last_name on employees (last_name);

create outline sv_pp
id '4A7CF117DBBA752D84D109077954A59E'
mode 0
as (
  query (
  -- For loop
   subquery (
     subquery (
       subquery (
```

```
          EMPLOYEES 1    access path index      EMP_LAST_NAME
              join by cross to
          JOB_HISTORY 0  access path index      JOB_HISTORY_HASH
          )
          union with
        subquery (
          CANDIDATES 2    access path sequential
          )
        )
      )
    )
   )
 compliance optional;

select cj.last_name, cj.first_name, cj.employee_id, cj.job_code,
       cj.department_code, cj.supervisor_id, cj.job_start
  from (select e.last_name, e.first_name, e.employee_id,
               jh.job_code, jh.department_code,
               jh.supervisor_id, jh.job_start
          from job_history jh, employees e
          where jh.employee_id = e.employee_id) cj
       left outer join
       (select last_name, first_name
          from candidates
          order by last_name, first_name) c
       on c.last_name = cj.last_name
       where cj.last_name is null;
```

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.1.12 JOIN Query with a Function in Predicate Bugchecks

Bug 3499717

The following INNER JOIN query with a function applied to one of the equi−join predicates bugchecks.

```
select count(*)
from
  (select xxxx_id, yyyy_id
     from  TAB1 where
          operation_no = '50300' and
          yyyy_loc between '00' and '99'
  ) as q1
inner join
  (select xxxx_id, product_id
     from  TAB2 where
          operation_no = '01300'
  ) as q2
on q1.xxxx_id = q2.xxxx_id and
   trim(q1.yyyy_id) = q2.xxxx_id ;
RDMS−I−BUGCHKDMP, generating bugcheck dump file DISK:[DIRECTORY]RDSBUGCHK.DMP;
***** Exception at 00FE8E18 : RDMS$$FIND_VALID_SEG_CRTV + 000000B8
```

As a workaround, the query works if the SQL flag 'NOMAX_SOLUTION' is set, as in the following example.

```
SQL> set flags 'nomax_solution';
```

```
  OR
$DEFINE RDMS$$DISABLE_MAX_SOLUTION 1

Here is the new strategy of the new run:

Tables:
  0 = TAB1
  1 = TAB2
Aggregate: 0:COUNT (*)
Cross block of 2 entries
  Cross block entry 1
    Merge of 1 entries
      Merge block entry 1
      Conjunct: 1.OPERATION_NO = '01300'
      Get     Retrieval sequentially of relation 1:TAB2
  Cross block entry 2
    Merge of 1 entries
      Merge block entry 1
      Conjunct: (0.XXXX_ID = 1.XXXX_ID) AND (TRIM (BOTH ' ' FROM 0.YYYY_ID) =
                1.XXXX_ID)
      Leaf#01 BgrOnly 0:TAB1 Card=10
        Bool: (0.OPERATION_NO = '50300') AND (0.YYYY_LOC >= '00') AND (
              0.YYYY_LOC <= '99')
        BgrNdx1 TAB1_IDX_S1 [2:2] Fan=9
          Keys: (0.XXXX_ID = 1.XXXX_ID) AND (0.YYYY_LOC >= '00') AND (0.YYYY_LOC
                 <= '99')
          Bool: 0.OPERATION_NO = '50300'

           0
1 row selected
```

This problem is caused by the fix made for Bug 1635351 in Oracle Rdb Release 7.0.6.3.

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.1.13 DIOBND$FETCH_AIP_ENT + 000001D4 Bugcheck

Bug 3360543

READ ONLY transactions sometimes caused a bugcheck in DIOBND$FETCH_AIP_ENT when the metadata required by the transaction was being manipulated by another process.

The bugcheck has been replaced by the following error message:

```
%RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-F-CANTFINDLAREA, cannot locate logical area nnn in area inventory page
list
```

This problem has been corrected in Oracle Rdb Release 7.1.2.4. The problem was that the READ ONLY transaction had knowledge of a now obsolete table or index logical area. Attempts to use that index or table partition now fail. When this problem occurs, the transaction should ROLLBACK and start a new transaction that will read the latest metadata from the database.

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.1.14 Various Bugchecks and Corruptions With Indexes of Type is Sorted Ranked

Bugs 3424257, 3289081, 3364208, 3424296 and 3013421

While adding or removing records in an index of *TYPE IS SORTED RANKED*, it was possible that one of several different bugchecks or corruptions could be observed.

Buchecks could occur that do not usually indicate index corruption including various bugchecks in the following routines:

- PSII2SPLITNODE
- PSII2REMOVEDUPBBC
- PSII2INSERTDUPBBC

Bugchecks could occur that usually do indicate index corruption including buchecks in the following routines:

- PSII2REMOVET
- PSIINDEX2GETSEPAFTER

In most cases, the corruption of the index would be reported by *RMU/VERIFY* as either an inconsistant cardinality or an invalid or negative amount of free space in a node.

The following example shows the types of errors that may be reported by *RMU/VERIFY* for a corrupt index.

```
%RMU−W−BTRLENERR, area RDB$SYSTEM, page 68, line 5
                 b−tree node length error
                 expected node length 58, found: 0
%RMU−I−BTRERPATH, parent B−tree node of 66:68:5 is at 66:18:3
...
%RMU−I−BTRENTLEN, B−tree node entry 1 has an invalid data length of 3345.
%RMU−I−BTRNODDBK, Dbkey of B−tree node is 210:5307:1
...
%RMU−W−BTRLEACAR, Inconsistent leaf cardinality (C2) of 1 specified
                  for entry 29 at dbkey 210:6196:0 using precision of 33.
```

Under rare circumstances, it was possible that when multiple records were deleted from the same key value in the same transaction, a *%RDB−E−NO_RECORD* error could be returned. A subsequent attempt to delete the same record would succeed without error. In this case, the dbkey being reported would be the dbkey of a duplicates overflow node in the index affected by the delete.

The following example shows the type of message returned. The dbkey is a dbkey of an index overflow node that collapsed and was deleted in a previous operation within the same transaction.

```
%RDB−E−NO_RECORD, access by dbkey failed because dbkey is no longer associated
with a record
−RDMS−F−NODBK, 66:71:0 does not point to a data record
```

If any index corruptions are detected they can be removed by dropping and recreating the offending index.

As a workaround, the problem can be avoided by using alternate index types.

This problem has been corrected in Oracle Rdb Release 7.1.2.4. Once this version is installed, it is strongly recommended that all indices of *TYPE IS SORTED RANKED* be verified using a command similar to *RMU/VERIFY/INDEX/DATA*. If any corruptions are reported, the index must be dropped and recreated to remove the corruption.

# 2.2 SQL Errors Fixed

## 2.2.1 Unexpected UNSDTPCVT Error Using DBKEY in UNION Clause

Bug 3426717

In a prior release of Oracle Rdb, the character set assignments for CASE expressions, UNION, EXCEPT, and INTERSECT operators were improved to allow similar character sets to be merged in the output CHAR and VARCHAR columns. Unfortunately, if the merge included a DBKEY column an error was reported.

The following example shows the unexpected error.

```
SQL> select '' from employees
cont> union all
cont> select dbkey from job_history;
%SQL-F-UNSDTPCVT, Unsupported data type conversion
```

This problem has been corrected in Oracle Rdb Release 7.1.2.4. SQL will again allow the merge of a DBKEY with character data. However, the resulting column value may not contain valid DBKEY values. Oracle recommends using a DBKEY literal instead of a character literal when merging DBKEY values.

```
SQL> select _dbkey'0:0:0' from employees
cont> union all
cont> select dbkey from job_history;
...
```

## 2.2.2 ACCVIO @ GEM_IP_BUILD + 3F7C from SQLMOD Compiler

Bug 3297277

If a SQL module had a singleton select statement such that one of the targets of the INTO list was a SQL function or an external function, SQL$MOD would fail and bugcheck. (Note that this is a syntax error since the targets of the INTO list in a singleton select may only be parameter or variable names.)

For example, consider the following SQL module language program which has an external function named EXT_FUNC as a target in the INTO list of the singleton select statement in procedure READ_FIELDS.

```
MODULE          BAD_FUNC_REF
LANGUAGE        GENERAL
PARAMETER       COLONS
--
PROCEDURE READ_FIELDS
   SQLCODE
   :EMP_ID      ID_DOM
   :LAST_NAME   LAST_NAME_DOM
   :FIRST_NAME  FIRST_NAME_DOM;
   SELECT
      LAST_NAME,
      FIRST_NAME
```

```
   INTO
      EXT_FUNC(:LAST_NAME),
      :FIRST_NAME
   FROM EMPLOYEES
   WHERE EMPLOYEE_ID = :EMP_ID;
```

When the above module was compiled, SQL$MOD would fail with the following message:

```
%RDMS-I-BUGCHKDMP, generating bugcheck dump file MY_DISK:[MY_DIR]SQLBUGCHK.DMP;
```

The generated bugcheck dump file contained the following failure:

```
***** Exception at 0015E92C : GEM_IP_BUILD + 00003F7C
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=0000000000000000, PC=000000000015E92C, PS=0000001B
```

The problem has been fixed so that compiling the above program now results in the following output:

```
      EXT_FUNC(:LAST_NAME),
      1
%SQL-F-INVINTOTAR, (1) target for an INTO clause must be a variable or parameter
```

As a workaround for this problem, remove the syntax error from the program (for example have only parameter or variable names as the target of an INTO list for a singleton select).

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.2.3 Builtin Function Names Can Not be Used as Undelimited Column Names

Bug 3513360

The following example shows that LENGTH can not be used as a column name now that Rdb includes a LENGTH builtin function. LENGTH is a synonym for CHAR_LENGTH.

```
SQL> create table dimensions (height int, length int, breadth int);
%SQL-I-DEPR_FEATURE, Deprecated Feature: Keyword BREADTH used as an identifier
SQL> select height from dimensions;
0 rows selected
SQL> select length from dimensions;
select length from dimensions;
              ^
%SQL-F-LOOK_FOR, Syntax error, looking for (, found FROM instead
SQL>
```

A workaround is to enable delimited identifier support using SET QUOTING RULES 'SQL99'.

```
SQL> set quoting rules 'sql99';
SQL> select "LENGTH" from dimensions;
0 rows selected
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.1.2.4 and includes corrections for the following functions: ABS, BITSTRING, GREATEST, LEAST, LENGTH, LENGTHB, NVL2, SIZEOF, STDDEV,

STDDEV_POP, STDDEV_SAMP, VARIANCE, VAR_POP, VAR_SAMP, and VSIZE.

### 2.2.4 REVOKE ENTRY No Longer Aborts on Missing User

Bug 2141456

In previous versions of Oracle Rdb, a wildcard REVOKE ENTRY statement might fail if the specified user was not present in some object's access control list.

The following example shows the problem.

```
SQL> GRANT ALL ON TABLE * TO SAMPLE_USER;
SQL>  --
SQL> REVOKE ENTRY ON TABLE JOBS FROM SAMPLE_USER;
SQL>  --
SQL> REVOKE ENTRY ON TABLE * FROM SAMPLE_USER;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-ACENOTFND, no matching access control entry found
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.1.2.4. With this release, a missing entry will no longer cause an error to be reported for REVOKE ENTRY.

### 2.2.5 SQLPRE71/CC Does Not Comment Out All EXEC SQL Statements

Bug 3078640

In some cases, the SQL Precompiler would fail to comment out an EXEC SQL statement. In particular, this would occur if the EXEC SQL statement was placed between the parameter type declarations of a K&R C−style function declaration and the opening "{" of the function. (Note that this is improper placement of an EXEC SQL statement.)

For example, consider the following C program fragment which contains embedded SQL:

```
EXEC SQL BEGIN DECLARE SECTION;
        EXEC SQL INCLUDE SQLCA;
        EXEC SQL INCLUDE SQLDA;

my_func(parm1, parm2)
char *parm1, *parm2;

EXEC SQL END DECLARE SECTION;

{ /* begin function */
...
} /* end function */
```

When the above program was processed by the SQL precompiler, the generated code ought to have "/*" and "*/" wrapped around the "EXEC SQL END DECLARE SECTION" statement but didn't. This resulted in the following diagnostic from the C compiler.

```
exec sql end declare section;
```

```
^
%CC−E−OPENBRACE, Missing "{".
at line number 119 in file MY_DISK:[MY_DIR]MY_PROG.C;1
```

This problem has been corrected so that the precompiler comments out the "EXEC SQL END DECLARE SECTION" statement.

As a workaround for this problem, locate the "EXEC SQL END DECLARE SECTION" statement before the function declaration.

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.2.6 SQLPRE/CC Loops When Program Contains Invalid C

Bug 2605882

A C structure type declaration which has a circular reference would cause the SQL Precompiler to run indefinitely with the CPU and memory usage gradually increasing until the process is either aborted or runs out of some resource.

For example, the following structure definition would exhibit these symptoms because the structure element "b" refers to the structure type being defined. (Note: this is not valid C code.)

```
    struct looper {short a; struct looper b};
```

This problem has been fixed so that the SQL Precompiler will detect the circular structure definition and process it properly. In addition, it will flag references in an EXEC SQL statement to C variables declared as such structures as shown in the following example:

```
    struct looper looper_cmd;

    exec sql select employee_id into :looper_cmd from employees
                                      1
%SQL−F−CIRCULARDEF, (1) Structure field b is a circular definition
```

As a workaround for this problem, fix the coding error so that the structure has a valid, non−selfreferential declaration.

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.2.7 Corrupt Storage Map Possible if String Literals Incorrectly Used for Numeric Columns

Bug 3530235

In prior releases of Oracle Rdb, it was permitted to create a storage map for an index or a table having the WITH LIMIT OF as character strings even though the column type was numeric. However, the resulting storage map was corrupted and would not allow rows to be inserted.

The following example shows this problem.

```
SQL> create database filename testdb
cont>    create storage area sa1;
SQL> create table t1 (col1 integer);
SQL> create storage map t1_map for t1
cont>   store using (col1)
cont>       in sa1 with limit of ('10');
%SQL-I-NUMCMPTXT, Numeric column will be compared with string literal as text
SQL> commit;
SQL> insert into t1 values (1);
%RDB-E-CONVERT_ERROR, invalid or unsupported data conversion
-RDMS-E-CSETBADASSIGN, incompatible character sets prohibit the requested
assignment
```

This problem may be observed when upgrading from an older Rdb release using SQL IMPORT. In this case, the solution is to include the redefinition of the STORAGE MAP in the IMPORT command so that the WITH LIMIT OF values are numeric.

This problem has been corrected in Oracle Rdb Release 7.1.2.4. The procedure which upgraded the character literals to numeric during CREATE STORAGE MAP and CREATE INDEX has been corrected to support literals with character set data.

## 2.2.8 Unclear Error When No Room for Compilation Command File

Bug 2690253

One may sometimes get a bugcheck dump with the following characteristics:

```
Alpha OpenVMS 7.2-2
Oracle Rdb Server 7.1.2.2.1
Got a SQLBUGCHK.DMP
COSI-F-WRITERR, write error
Exception occurred at SQL$$FAT_OPERATION_ERROR + 00007054
Called from SQL$$SUBMIT_COM_FILE + 00000660
Called from SQL$$SUBMIT_COM_FILE + 00000278
Running image SQL$MOD70.EXE
```

Such a bugcheck showed that there has been an error doing a write but gave no details about the file involved. For example, the bugcheck might be produced when the SQL Precompiler has a problem writing a command procedure to call a compiler to process the precompiler output. (The file is written to SYS$SCRATCH.) A likely error in this case was that the device was full.

An additional message has been added to the bugcheck output to identify the file involved in the failure. This message appears immediately after the line with the COSI_F_WRITERR message and appears similar to the following:

```
%SQL-F-UNBWRTFIL, Unable to write to file SYS$SCRATCH:MY_COMFIL.COM
```

As a workaround, have Oracle Rdb Support use the symbolic references in the bugcheck dump to analyze the Rdb code and determine what file was associated with the failure.

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.2.9 Unexpected SQL−F−QUETOOBIG Error During CREATE TABLE

Bug 3542644

In prior versions of Oracle Rdb, a CREATE TABLE or ALTER TABLE statement with many constraints, DEFAULT, AUTOMATIC AS, or COMPUTED BY definitions could fail with the following error.

```
%SQL−F−QUETOOBIG, Query or routine contains too many table references
```

A workaround for this problem is to split the CREATE TABLE or ALTER TABLE into small ALTER TABLE statements.

This problem has been corrected in Oracle Rdb Release 7.1.2.4. SQL now resets the counter for each constraint, default value and COMPUTED BY expression allowing many more definitions in a single command.

## 2.2.10 Performance and Limits Problems with Concatenation Operator

Bugs 2641023 and 3539378

In prior versions of Oracle Rdb, the CONCAT or || operator could cause excessive resource usage when used in a query under the ORACLE LEVEL1 or ORACLE LEVEL2 dialects, such as through SQL*net for Rdb. Reported problems included:

- Query fails with BUFLIMEXC errors

  ```
  %RDB−E−IMP_EXC, facility−specific limit exceeded
  −RDMS−F−BUFLIMEXC, internal buffer limit exceeded
  ```
- Query incurs excessive page faults during execution

These problems were caused by overly complex query generation required to implement Oracle semantics for NULL and zero length string usage for the concatenation operator. Oracle RDBMS semantics require that concatenation treat NULL as a zero length string, and conversely that VARCHAR function results (TRIM, SUBSTRING, RTRIM, TO_CHAR, etc) have zero length treated identically with NULL.

These problems has been corrected in Oracle Rdb Release 7.1.2.4. When concatenation is mixed with VARCHAR string functions or other concatenation operators, further optimizations are performed to minimize the complexity of the generated query. Note that this problem did not occur with any other dialect.

## 2.2.11 Unexpected BAD_REQ_HANDLE Error After Failed Query Compile

Bug 3536062

In previous versions of Oracle Rdb, a failed compile of a query that used nested stored procedures or functions could cause subsequent attempts to execute these nested routines to fail. The compile might fail due to any number of errors: a lock timeout while reading metadata, an invalidated stored routine, obsolete metadata, and so on.

The following example shows the reported error from Rdb. Even after a ROLLBACK of the statement that caused the OBSOLETE_METADATA error, the routine still fails with a RTN_FAIL and BAD_REQ_HANDLE error.

```
SQL> set flags 'trace,warn_invalid';
SQL>
SQL> drop table TTT cascade;
~Xw: Routine "C" marked invalid
SQL>
SQL> begin call A(); end;
%RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-F-BAD_SYM, unknown relation symbol - TTT
SQL> begin call A(); end;
~Xt: in A
~Xt: in B
%RDB-E-RTN_FAIL, routine "(unknown)" failed to compile or execute successfully
-RDB-E-BAD_REQ_HANDLE, invalid request handle
SQL>
SQL> rollback;
SQL>
SQL> begin call A(); end;
~Xt: in A
~Xt: in B
%RDB-E-RTN_FAIL, routine "(unknown)" failed to compile or execute successfully
-RDB-E-BAD_REQ_HANDLE, invalid request handle
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.1.2.4. Rdb now correctly reprocesses the nested calls after a failure during the request compile.

## 2.2.12 EXCEPT, INTERSECT and MINUS Operators May Cause Query to Fail

Bug 3586361

In prior releases of Rdb, queries that contained more than two EXCEPT, MINUS or INTERSECT operators would generate a bugcheck.

The following example shows such a query and the resulting bugcheck.

```
SQL> select col1 from tab1
cont> except
cont> select col1 from tab2
cont> except
cont> select col1 from tab3;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file
    DISK1:[RDB_BUGCHECK]RDSBUGCHK.DMP
%RDMS-I-BUGCHKDMP, generating bugcheck dump file
    DISK1:[RDB_BUGCHECK]SQLBUGCHK.DMP
%SYSTEM-F-ACCVIO, access violation, reason mask=04, virtual address=
0000000000000000, PC=00000000003F1DAC, PS=0000001B
```

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

# 2.3 RMU Errors Fixed

## 2.3.1 RMU/RECOVER/ONLINE Without /JUST_CORRUPT or /AREA Qualifiers Bugchecks

As described in the documentation, when issuing an RMU/RECOVER command, the /ONLINE qualifier can only be use in conjunction with the /JUST_PAGE or /AREA qualifiers.

In previous releases, if the /ONLINE qualifier was used alone, it would result in an RMU bugcheck as shown below.

```
$ RMU/RECOVER/LOG/ONLINE AIJ_BCK.AIJ
%RMU−E−RECFAILED, fatal, unexpected roll−forward error detected at AIJ record 1
%COSI−F−BUGCHECK, internal consistency failure
%RMU−F−FATALOSI, Fatal error from the Operating System Interface.
%RMU−I−BUGCHKDMP, generating bugcheck dump file disk:[dir]RMUBUGCHK.DMP;
%RMU−F−FTL_RCV, Fatal error for RECOVER operation at <timestamp>
```

The exception and the call frame in the dump are as follows.

```
***** Exception at 007212B8 : KUTREC$ABORT + 000005D8
%COSI−F−BUGCHECK, internal consistency failure
Saved PC = 0071F5E4 : KUTREC$SETUP + 00000214
Saved PC = 00474198 : RMUREC$RECOVER_ACTION + 00000948
Saved PC = 004737D8 : RMUCLI$RECOVER + 000004C8
Saved PC = 003A56F4 : RMU_DISPATCH + 00000D44
Saved PC = 003A4508 : RMU_STARTUP + 000004A8
Saved PC = 001E002C : RMU$MAIN + 0000002C
```

Now RMU will detect missing /JUST_PAGE or /AREA qualifiers when /ONLINE is used and will return an error message.

```
$ RMU/RECOVER/LOG/ONLINE AIJ_BCK.AIJ
%RMU−F−CONFLSWIT, conflicting qualifiers /ONLINE and /AREAS or /JUST_CORRUPT
missing
%RMU−F−FTL_RCV, Fatal error for RECOVER operation at <timestamp>
```

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.3.2 Invalid RMU−E−INASPAREA Message During RMU/BACKUP or RMU/VERIFY

Bug 3424732

At the start of an RMU/BACKUP for Rdb V7.1, the database root is verified. While doing the verify of the database root, each live storage area entry in the database root was correctly reloaded to get its current state on all cluster nodes accessing the database but the corresponding snapshot area entry was not reloaded by RMU/BACKUP to bring it up to date. If a new database storage area had just been created on one node in the cluster, this could lead to the following incorrect RMU/BACKUP error message being output on another cluster node which was doing an RMU/BACKUP of the same database.

```
RMU−E−INASPAREA, Live area AREA_NAME points to a snapshot area that is
inactive.
```

Repeating the RMU/BACKUP would not show this error since this would reload the snapshot entry and bring it up to date. This problem has been fixed.

Note that this same problem could also happen if an RMU/VERIFY was being done instead of an RMU/BACKUP since both share the same code to verify the database root.

The following example shows the sequence of adding a storage area entry to a database on one cluster node and then backing up the database on another node that gave the RMU−E−INASPAREA error.

On one cluster node...

```
$ @RDM$DEMO:PERSONNEL SQL M NOCDD
$ SQL
SQL> ALTER DATABASE FILENAME MF_PERSONNEL OPEN IS MANUAL;
SQL> ALTER DATABASE FILENAME MF_PERSONNEL RESERVE 1 STORAGE AREA;
SQL> EXIT;
$ RMU/BACKUP MF_PERSONNEL
```

On another node in the cluster...

```
$ RMU/OPEN MF_PERSONNEL
```

Back to first cluster node...

```
$ SQL
SQL> ALTER DATABASE FILENAME MF_PERSONNEL ADD STORAGE AREA TEST;
SQL> EXIT;
```

Back to the second cluster node...

```
$ RMU/BACKUP MF_PERSONNEL
%RMU−E−INASPAREA, Live area TEST points to a snapshot area that is inactive.
```

The workaround for this problem is to do the RMU/BACKUP or RMU/VERIFY on the same cluster node where the storage area was created or to repeat the RMU/BACKUP or RMU/VERIFY on the second cluster node if the RMU−E−INASPAREA error is returned on the first RMU/BACKUP or RMU/VERIFY.

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.3.3 RMU/BACKUP/AFTER Returned Successful Status Despite Getting a Previous Error

Bug 3168647

In previous releases of Rdb, it was possible under certain conditions, such as AIJ file marked inaccessible, that RMU/BACKUP/AFTER returned a successful status even if a fatal error had been encountered, as shown below (see lines marked [**]).

```
$ RMU/BACKUP/AFTER/LOG DB FOO_AIJ_BCK.AIJ
```

```
     %RMU-I-AIJBCKBEG, beginning after-image journal backup operation
     %RMU-I-OPERNOTIFY, system operator notification: AIJ backup operation
        started
     %RMU-I-AIJBCKSEQ, backing up after-image journal sequence number 0
     %RMU-I-LOGBCKAIJ, backing up after-image journal FOO_AIJ1 at 05:11:02.97
[**] %RMU-F-AIJBCKINAC, AIJ backup completed when accessing inaccessible
        journal FOO_AIJ1
     %RMU-I-AIJNOBACKUP, AIJ contains no transactions that qualify for backup
     %RMU-I-OPERNOTIFY, system operator notification: AIJ backup operation
        completed
[**] %RMU-I-AIJBCKEND, after-image journal backup operation completed
        successfully
     $ sh symb $STATUS
       $STATUS == "%X10000001"
```

Now, when an error is encountered, RMU/BACKUP/AFTER will raise an exception and a failure status will be returned as shown below.

```
$ RMU/BACKUP/AFTER/LOG DB FOO_AIJ_BCK.AIJ
%RMU-I-AIJBCKBEG, beginning after-image journal backup operation
%RMU-I-OPERNOTIFY, system operator notification: AIJ backup operation started
%RMU-I-AIJBCKSEQ, backing up after-image journal sequence number 0
%RMU-I-LOGBCKAIJ, backing up after-image journal FOO_AIJ1 at 07:45:53.58
%RMU-I-AIJBCKSTOP, backup of after-image journal FOO_AIJ1 did not complete
%RMU-I-OPERNOTIFY, system operator notification: AIJ manual backup operation
    failed
%RMU-F-AIJBCKINAC, AIJ backup completed when accessing inaccessible journal
    FOO_AIJ1
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at <timestamp>
$ sh symbol $STATUS
  $STATUS == "%X12C8AE3C"
```

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.3.4 Database Recovery Caused by RMU/BACKUP/ONLINE Lock Timeout

Bug 3388349

There was an error handling problem in RMU/BACKUP/ONLINE. When a lock timeout was signaled, RMU/BACKUP did not properly unbind from the database being backed up. This caused the Rdb monitor to properly launch a recovery operation for the database. In Oracle Rdb Release 7.1.2.4, this has been changed so the unbind from the database will occur when the lock timeout is signaled and the database recovery will not be necessary and will not take place. Note that this problem did not cause any corruption to the database being backed up.

The following example shows that when a lock timeout was signaled during an RMU/BACKUP/ONLINE, a database recovery took place as shown in the Rdb monitor log.

```
$RMU/BACKUP/ONLINE/QUIET_POINT/LOCK_TIMEOUT=10 MF_PERSONNEL MFP.RBF
%RMU-F-TIMEOUT, timeout on quiet
-COSI-W-CANCEL, operation canceled
%RMU-F-FTL_BCK, Fatal error for BACKUP operation at 12-JAN-2004 16:31:09.35
```

The Rdb monitor log showed that a database recovery took place.

```
$type SYS$SYSTEM:RDMMON71.LOG

12−JAN−2004 16:31:09.35 − Received user  image termination from 25E0103D:1
 − database name "DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1" [_DSA6002] (36,391,0)
 − abnormal user termination detected
 − database monitor created Database Recovery process RDM_RB71_1 (25E02649)
 − user termination suspended until recovery ready
```

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.3.5 Incorrect Verification of Invalid Reference DBKEYs in Ranked Indices

Bugs 3009262 and 3532702

If an index of *TYPE IS SORTED RANKED* had invalid reference dbkeys in duplicate overflow nodes, RMU/VERIFY/INDEX could fail to report the error. If the error was reported, it was reported as an informational message.

This type of corruption was possibly introduced by the problem referenced in Section 3.1.13.

In the following example, the index corruption is correctly reported.

```
$ RMU/VERIFY/INDEX=MY_IDX MY_DB
%RMU−E−BADREFDBK, Invalid reference pointer 358:263034:0 for duplicate B−tree
node 299:88538:0
%RMU−I−DUPRECDBK, the last duplicate record dbkey was 358:294862:1
%RMU−I−DUPOWNDBK, Dbkey of owner of this duplicate node is 299:89267:1
%RMU−I−BTRERPATH, parent B−tree node of 299:89267:1 is at 299:89245:0
%RMU−I−BTRERPATH, parent B−tree node of 299:89245:0 is at 299:89138:1
%RMU−I−BTRERPATH, parent B−tree node of 299:89138:1 is at 299:88639:1
%RMU−I−BTRERPATH, parent B−tree node of 299:88639:1 is at 299:88437:1
%RMU−I−BTRROODBK, root dbkey of B−tree is 299:88437:1
```

The problem represents a real corruption in the index and can lead to bugchecks as described in Section 3.1.13.

To avoid the problem, the offending index must be dropped and recreated.

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.3.6 RMU/ANALYZE/TRANSACTION_TYPE=READ_ONLY Could Hold Quiet Point Lock in CR Mode

Bug 3456640

The position of the /TRANSACTION_TYPE qualifier during an RMU/ANALYZE command caused the quiet point lock to be held in CR mode for the following case:

```
$RMU/ANALYZE/INDEX/OPTION=DEBUG  SQL$DATABASE  /TRANSACTION=READ_ONLY
```

However, the quiet point lock was held in NL mode for the following case:

```
$RMU/ANALYZE/INDEX/OPTION=DEBUG/TRANSACTION=READ_ONLY  SQL$DATABASE
```

This could cause delay in an RMU/BACKUP/ONLINE which waits to acquire the quiet point lock. Now, in both these cases, the quiet point lock will be correctly held in NL mode.

The following example shows that if the /TRANSACTION_TYPE=READ_ONLY qualifier was placed after the database name parameter instead of before the database name parameter in an RMU/ANALYZE command, the quiet point lock was held in CR mode during the RMU/ANALYZE operation instead of in the correct NL mode.

In the first process:

```
$RMU/ANALYZE/INDEX/OPTION=DEBUG  SQL$DATABASE  /TRANSACTION=READ_ONLY
```

In another process, display the locks for the first process while the RMU/ANALYZE executes.

```
$RMU/SHOW LOCKS/PROCESS=2060FEEE

================================================================================
SHOW LOCKS/PROCESS Information
================================================================================
--------------------------------------------------------------------------------
Resource: quiet

          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Owner:    2060FEEE  Proc First 1..      54004849  00010003            CR
```

The workaround for this problem is to place the /TRANSACTION_TYPE=READ_ONLY qualifier before the database parameter.

In the first process:

```
$RMU/ANALYZE/INDEX/OPTION=DEBUG/TRANSACTION=READ_ONLY  SQL$DATABASE
```

In another process, display the locks for the first process while the RMU/ANALYZE executes.

```
$RMU/SHOW LOCKS/PROCESS=2060FEEE

================================================================================
SHOW LOCKS/PROCESS Information
================================================================================
--------------------------------------------------------------------------------
Resource: quiet

          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Owner:    2060FEEE  Proc First 1..      54004849  00010003            NL
```

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.3.7 RMU/BACKUP/AFTER Hangs with Single Extensible AIJ File

Bugs 3528947 and 3531179

Starting with Oracle Rdb Release 7.1.2.3, it was possible to have RMU/BACKUP/AFTER hang when using a single extensible AIJ file. Actually there was a deadlock between the backup process and the users.

Under such a condition, RMU/SHOW STATISTICS/SCREEN="STALL MESSAGE" will look like the following.

```
28C03AA0:1  26-MAR-2004 07:30:24.94 - waiting for AIJ journal open 0 (CR)
28C03AA5:1u 26-MAR-2004 07:30:24.95 - waiting for AIJ journal info 0 (EX)
```

The RMU/SHOW LOCK/MODE=BLOCKING will show:

```
--------------------------------------------------------------------------------
Resource: AIJ journal info 0
          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Waiting:  28C03AA5  BCK............     3300FDF6  00010046  EX        NL
Blocker:  28C03AA0  USR............     6A033A29  00010046            CR
--------------------------------------------------------------------------------
Resource: AIJ journal open 0
          ProcessID Process Name        Lock ID   System ID Requested Granted
          --------- ---------------     --------- --------- --------- -------
Waiting:  28C03AA0  USR............     38034203  00010046  CR
Blocker:  28C03AA5  BCK............     2A03E8AF  00010046            EX
--------------------------------------------------------------------------------
```

Since this only applied to a single extensible AIJ file, a possible workaround is to use multiple circular AIJs.

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

## 2.3.8 RMU Command to Change OVERWRITE Flag is an Offline Command

Bugs 613934 and 2101705

As documented in the RMU Reference Manual, the modification of the OVERWRITE flag is an offline command. It can be done only if no users are attached to the database.

In previous releases of Rdb, RMU would allow you to change the OVERWRITE flag even if users were attached to the database while SQL would return an error message, as shown below.

```
$ RMU/DUMP/USER FOO
Active user with process ID 30C5064A
    Stream ID is 1
    Monitor ID is 1 (<system>)
    Transaction ID is 11
    No transaction in progress
    Last Process quiet-point was AIJ sequence 0
$
```

```
$ RMU/SET AFTER/OVERWRITE FOO
$
$ SQL$
SQL> alter data file foo
cont> journal enable (overwrite enable);
%RDMS−F−NOEUACCESS, unable to acquire exclusive access to database
SQL>
```

Now an error message will be returned by RMU if you try to change the OVERWRITE flag when users are attached to the database, as shown below:

```
$ RMU/DUMP/USER FOO
Active user with process ID 30C5064A
    Stream ID is 1
    Monitor ID is 1 (<system>)
    Transaction ID is 11
    No transaction in progress
    Last Process quiet−point was AIJ sequence 0
$
$ RMu/SET AFTER/OVERWRITE FOO
%RMU−I−WAITOFF, Waiting for offline access to FOO
```

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

# 2.4 Hot Standby Errors Fixed

## 2.4.1 Sequences Not Replicated by Hot Standby

Bug 3466579

When sequences were being used, they were not properly propogated to the standby database by the Hot Standby feature.

```
$ SQL$
SQL> ATTACH 'FILENAME [.MASTER]MF_PERSONNEL';
SQL> CREATE SEQUENCE TEST;
SQL> COMMIT;
SQL> DISCONNECT ALL;
SQL> $ WAIT 0:0:10
SQL> ATTACH 'FILENAME [.STANDBY]STANDBY_PERSONNEL';
SQL> SHOW SEQUENCE
Sequences in database with filename [.STANDBY]STANDBY_PERSONNEL
%RDMS−I−BUGCHKDMP, generating bugcheck dump file DEV:[DIR]RDSBUGCHK.DMP;
%COSI−F−BUGCHECK, internal consistency failure
$
```

The bugcheck would contain an exception similar to the following:

```
***** Exception at 0129EE00 : SEQ$GET_SEQUENCE + 000000E0
%COSI−F−BUGCHECK, internal consistency failure
```

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.2.4.

# Chapter 3
# Software Errors Fixed in Oracle Rdb Release 7.1.2.3

This chapter describes software errors that are fixed by Oracle Rdb Release 7.1.2.3.

# 3.1 Software Errors Fixed That Apply to All Interfaces

## 3.1.1 RMU/CLOSE/CLUSTER Gets ALSACTIVE Error After Replication Stopped

Bug 2618430

After issuing the ***RMU/REPLICATION AFTER STOP/WAIT*** command, an ***RMU/CLOSE/CLUSTER*** command would sometimes return an ALSACTIVE error indicating that replication was still active. For example:

```
$ RMU/REPLICATE AFTER_JOURNAL STOP /WAIT MASTER_DB
%RMU-I-HOTSTOPWAIT, stopping database replication, please wait
$ RMU/CLOSE/CLUSTER MASTER_DB
%RDMS-F-CANTCLOSEDB, database could not be closed as requested
-RDMS-F-ALSACTIVE, Database replication is active
%RMU-W-CANTOCDB, Error encountered while opening or closing database file
MASTER_DB
```

This error would occur because the ***RMU/REPLICATE*** command would occasionally return to the DCL prompt before all ALS servers on all nodes accessing the database had stopped replication.

Retrying the ***RMU/CLOSE*** command would usually succeed.

This problem has been corrected in Oracle Rdb Release 7.1.2.3. The ***RMU/REPLICATE*** command will not return until all ALS servers have stopped replication.

## 3.1.2 Enhancements to SET FLAGS CHRONO_FLAG

Enhancements 3109294 and 3109309

This release of Oracle Rdb includes the following changes to the CHRONO_FLAG.

- The obsolete keyword CRONO_FLAG has been removed. Please use CHRONO_FLAG.
- CHRONO_FLAG(0) and NOCHRONO_FLAG are equivalent.
- CHRONO_FLAG and CHRONO_FLAG(1), as in prior versions, enable an additional trace message which includes the attach number and the current time.
- CHRONO_FLAG(2) enables an additional trace message which includes the attach number and the current date and time.
- Any values included in the CHRONO_FLAG greater than 2 are currently ignored and a value 1 is assumed (i.e. only time is displayed).
- The CHRONO_FLAG is now applied to all trace messages output by the SET FLAGS TRANSACTION_PARAMETER option.

The following example shows the new output.

```
SQL> set flags 'chrono_flag(2),transaction';
SQL> start transaction;
 ATTACH #1, 29-NOV-2003 10:08:37.51
~T Compile transaction (1) on db: 1
```

```
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_WRITE (read write)
 ATTACH #1, 29-NOV-2003 10:08:37.58
~T Start_transaction (1) on db: 1, db count=1
SQL> rollback;
 ATTACH #1, 29-NOV-2003 10:08:46.74
~T Rollback_transaction (1) on db: 1
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.1.3 Wrong Sort Order for Query With Aggregate, Group By, Order By

Bug 3077857

Before the problem was fixed, the query in the example below returned the results in ascending order of the NTS_FLAG column, in contradiction to the explicit DESCENDING attribute in the ORDER BY clause. The problem appeared when RDM$BIND_BUFFERS was set to 200 but not when set to 20.

```
set flags 'detail,strategy';

select  Max(SERVICE_SUM.NTS_FLAG) NTS_FLAG
  from  SERVICE_SUM, INFO_PROVIDER_CODE_PARENT
  where SERVICE_SUM.CALL_END_DATE between
        DATE ANSI '2003-07-16' and  DATE ANSI '2003-07-16'
        and
        SERVICE_SUM.DIALED_NO = INFO_PROVIDER_CODE_PARENT.DIALED_NO
  group by SERVICE_SUM.NTS_FLAG
  order by SERVICE_SUM.NTS_FLAG DESC;
```

The change in the value for RDMS$BIND_BUFFERS had the effect of changing the optimizer query strategy from a CROSS join to a MATCH join. Using interactive SQL, for example, this could be seen by first enabling output of the optimizer strategy (see the SET FLAGS statement in the preceding example). The combination of MATCH join, aggregate query, ascending GROUP BY sort, and descending ORDER BY Sort, plus the optimizer logic to try to eliminate unnecessary sorts were the conditions needed for the problem to appear.

As a workaround, the problem could be made to go away by reducing the value in RDM$BIND_BUFFERS from 200 to 20 or by creating a query outline forcing a CROSS join strategy.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.1.4 Left Outer Join Query With SUBSTRING and CHAR_LENGTH Bugchecks

Bug 2851595

The following left outer join query with SUBSTRING and CHAR_LENGTH bugchecks during the process of creating a match retrieval strategy.

```
select e.employee_id
 from employees e left outer join job_history jh
 on (e.employee_id =
     substring (jh.employee_id from 1
                 for char_length (e.employee_id)));
%DEBUG−I−DYNMODSET, setting module RDMS$PREEXEMSC
%SYSTEM−F−BREAK, breakpoint fault at PC=003A4CD6, PSL=03C00004
break on exception at RDMS$PREEXEMSC\RDMS$$FIND_MEMBER_EQV_CLASS\%LINE 4486
DBG> SH CA
 module name    routine name     line                rel PC           ab
*RDMS$PREEXEMSC RDMS$$FIND_MEMBER_EQV_CLASS
*RDMS$PREEXE    RDMS$$SETUP_SORT32
*RDMS$PREEXE_CREATE
                RDMS$$CREATE_RSS$MTCH
```

The bugcheck is caused by a match order using equi−join keys of the outer join query, created from the following predicate:

```
 (e.employee_id = substring (jh.employee_id from 1
                             for char_length (e.employee_id))
```

Where the match key on the right side depends on its own context "employees e" from its left side.

As a workaround, the query works if the function CHAR_LENGTH is applied with JH table:

```
select e.employee_id
 from employees e left outer join job_history jh
 on (e.employee_id =
     substring (jh.employee_id from 1
                 for char_length (jh.employee_id)));
Tables:
  0 = EMPLOYEES
  1 = JOB_HISTORY
Match    (Left Outer Join)
  Outer loop
    Index only retrieval of relation 0:EMPLOYEES
      Index name  EMP_EMPLOYEE_ID [0:0]
  Inner loop
    Temporary relation
    Sort: SUBSTRING (1.EMPLOYEE_ID FROM (1 − 1) FOR CHAR_LENGTH (1.EMPLOYEE_ID))
          (a)
    Index only retrieval of relation 1:JOB_HISTORY
      Index name  JH_EMPLOYEE_ID [0:0]
 E.EMPLOYEE_ID
 00164
 00164
 ...etc...

274 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.1.5 Errors Not Reported by SQL

Bug 3083552

In some cases, when a bugcheck was produced, the SQLCODE returned was 0 indicating that no error had occurred.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.1.6 Partitioning Clause Not Working in Embedded SQL

Bug 3112810

In some cases, a sequential scan on partitioned data, where the partitioning uses a VARCHAR type column, can fail (or possibly produce wrong data results).

The following example shows a typical error message:

```
char a_alert[6] = "XYZ05";

EXEC SQL
    SET TRANSACTION READ WRITE WAIT RESERVING
        T_LIMIT PARTITION (3) FOR EXCLUSIVE READ;

EXEC SQL
    SELECT count(*)
    into   :i
    FROM   T_LIMIT
    WHERE  A_ALERT = :a_alert
      and  (
            EXTRACT (YEAR FROM A_DATE_CREATED) * 10000 +
            EXTRACT (MONTH FROM A_DATE_CREATED) * 100 +
            EXTRACT (DAY FROM A_DATE_CREATED)
           ) = 20031017;

%RDB-E-UNRES_REL, relation T_LIMIT in specified request is not a relation
     reserved in specified transaction
-RDMS-E-UNRES_AREA, area 67 within relation "T_LIMIT" is not reserved
```

Rdb is expecting the type of the parameter in the WHERE clause to agree with the type of the column it is comparing against (which is CHAR(5) in this case).

Interactive SQL does the data conversion and comparison correctly but SQL$PRE/CC sometimes gets confused.

Possible workarounds include:

- Use a compound statement to convert the type from VARCHAR to CHAR.
  Here we recode the example query (above) as a compound statement which assigns the input parameter into an internally declared CHAR(5) variable as follows:

  ```
  EXEC SQL
      BEGIN declare :my_char char(5);
      set :my_char = :a_alert;
      SELECT count(*)
      into   :i
      FROM   T_LIMIT
      WHERE  A_ALERT = :my_char
        and  (
  ```

```
          EXTRACT (YEAR FROM A_DATE_CREATED) * 10000 +
          EXTRACT (MONTH FROM A_DATE_CREATED) * 100 +
          EXTRACT (DAY FROM A_DATE_CREATED)
         ) = 20031017; END;
```

The above statement uses the "MY_CHAR" variable to convert the type of the input from VARCHAR (which is the only way that SQL$PRE/CC will send it) into CHAR.

* Use Dynamic SQL to avoid host variable inputs.
  For example, in the reproducer program declare SQLDA areas as follows:

```
EXEC SQL INCLUDE SQLDA;
...
    struct SQLDA_STRUCT *OUT_AREA;
    OUT_AREA = calloc(1, sizeof(struct SQLDA_STRUCT) +
                      (10*sizeof(struct SQLVAR_STRUCT)) );
    OUT_AREA->SQLN = 10;
```

  Then recode the query as a dynamic statement with no input parameters as in the following example:

```
    sprintf(sql_stmt,
            "SELECT count(*) \
            into   ? \
            FROM   T_LIMIT \
            WHERE  A_ALERT = '%s' \
              and  ( \
                    EXTRACT (YEAR FROM A_DATE_CREATED) * 10000 + \
                    EXTRACT (MONTH FROM A_DATE_CREATED) * 100 + \
                    EXTRACT (DAY FROM A_DATE_CREATED) \
                   ) = 20031017", a_alert);
    EXEC SQL PREPARE stmt1 from :sql_stmt;
    EXEC SQL DESCRIBE stmt1 OUTPUT into :OUT_AREA;
    OUT_AREA->SQLVAR[0].SQLDATA = (char *)&i;
    EXEC SQL EXECUTE stmt1 into descriptor :OUT_AREA;
```

* Use Dynamic SQL to force host variable parameters input to CHAR in an input SQLDA. This method has the advantage that the request can be reused. However, it is imperative that the input strings be space padded to the length of the data (five in our examples).
  For example, declare the SQLDA items as in the Workaround 2 example above plus the following:

```
    struct SQLDA_STRUCT *IN_AREA;
    IN_AREA = calloc(1, sizeof(struct SQLDA_STRUCT) +
                     (10*sizeof(struct SQLVAR_STRUCT)) );
    IN_AREA->SQLN = 10;
```

  Then recode the query to use an input SQLDA as follows:

```
    strcpy(sql_stmt, "SELECT count(*) \
            into   ? \
            FROM   T_LIMIT \
            WHERE  A_ALERT = ? \
              and  ( \
                    EXTRACT (YEAR FROM A_DATE_CREATED) * 10000 + \
                    EXTRACT (MONTH FROM A_DATE_CREATED) * 100 + \
                    EXTRACT (DAY FROM A_DATE_CREATED) \
                   ) = 20031017");
```

```
EXEC SQL PREPARE stmt2 from :sql_stmt;
EXEC SQL DESCRIBE stmt2 OUTPUT into :OUT_AREA;
EXEC SQL DESCRIBE stmt2 INPUT into :IN_AREA;
IN_AREA−>SQLVAR[0].SQLTYPE = SQLDA_CHAR;
IN_AREA−>SQLVAR[0].SQLDATA = a_alert;
OUT_AREA−>SQLVAR[0].SQLDATA = (char *)&i;

EXEC SQL EXECUTE stmt2 into descriptor :OUT_AREA
    using descriptor :IN_AREA;
```

In this example, the second DESCRIBE sets IN_AREA−>SQLVAR[0].SQLTYPE to SQLDA_VARCHAR. But overriding this value to SQLDA_CHAR causes dynamic SQL to build a request that works. Again please note that when using SQLDA_CHAR, the data string (a_alert in the example) must be space padded to the length specified in IN_AREA−>SQLVAR[0].SQLLEN.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.1.7 Join Query With GROUP_BY/ORDER_BY Returns Wrong Order

Bug 2851708

The following query with GROUP_BY/ORDER_BY, joining two derived tables with similar GROUP_BY clauses, returns the wrong order (should return in descending order).

```
set flags 'strategy, detail';
select  t1.name, t2.name, t1.datum, t2.datum
  from (select name, datum from a
          group by name, datum) t1
       join
       (select name, datum from b
          group by name, datum) t2
       on (t1.name = t2.name and t1.datum = t2.datum)
   group by t1.name, t2.name, t1.datum, t2.datum
   order by t1.name desc, t1.datum desc ;
Tables:
  0 = A
  1 = B
Reduce: 0.NAME, 0.DATUM, 1.NAME, 1.DATUM
Cross block of 2 entries
  Cross block entry 1
    Merge of 1 entries
      Merge block entry 1
      Reduce: 0.NAME, 0.DATUM
      Sort: 0.NAME(a), 0.DATUM(a)           <== See Note:
      Get Retrieval sequentially of relation 0:A
  Cross block entry 2
    Merge of 1 entries
      Merge block entry 1
      Reduce: 1.NAME, 1.DATUM
      Sort: 1.NAME(d), 1.DATUM(d)
      Conjunct: (0.NAME = 1.NAME) AND (0.DATUM = 1.DATUM)
      Get     Retrieval sequentially of relation 1:B
 T1.NAME   T1.DATUM
 AAAA      1−JAN−2000 00:00:00.00
```

```
 BBBB      1-JAN-2000 00:00:00.00
2 rows selected
```

This problem occurs when the main query contains a GROUP BY clause on the columns of the two joined derived tables with GROUP BY with the ORDER BY clause referencing the columns of the first table using descending order.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.1.8 Query Joining Two Derived Tables of View With UNION Overflows Stack

Bug 3194445

The following query, joining two derived tables of a view with UNION, overflows the stack.

```
select
  (select period_start_date from gl_period_vw
     where
       company_no = glpv.company_no and
       fiscal_yr = glpv.fiscal_yr
        )
     as p2_period_start_date,
  (select period_end_date from gl_period_vw
     where
       company_no = glpv.company_no and
       fiscal_yr = glpv.fiscal_yr
        )
     as p2_period_end_date
from gl_period_vw glpv, gl_period glp
;
%RDB-F-IMP_EXC, facility-specific limit exceeded
-RDB-I-TEXT, internal error -- query solution not found
```

The view is defined as:

```
create view GL_PERIOD_VW
    (COMPANY_NO,
     FISCAL_YR,
     PERIOD_START_DATE,
     PERIOD_END_DATE) as
    select
        C2.COMPANY_NO,
        C2.FISCAL_YR,
        C2.PERIOD_START_DATE,
        C2.PERIOD_END_DATE
    from GL_PERIOD C2
        where (C2.PERIOD_NO = 0)
    union all
    select
        C3.COMPANY_NO,
        C3.FISCAL_YR,
        C3.PERIOD_START_DATE,
        C3.PERIOD_END_DATE
```

```
    from GL_PERIOD C3, INTEGER_LIST C4
        where (C3.PERIOD_NO = 0)
    ;
```

This problem is caused by the fix made for Bug 2649215 where a query before bugchecks.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.1.9 Query With Shared Expression in Two Predicates Returns Wrong Result

Bug 3201864

The following query with a shared expression in two predicates should return one row but instead returns two rows.

```
set flags 'strategy,detail';

SELECT  T1.CODE,T2.ALUM, T1.ALUM, T2.CODE
  FROM T1, T2, T3 WHERE
          T2.TUNE    = T1.TUNE          AND
          T2.SLAM    = T1.SLAM          AND
          T2.STAR    = T1.STAR          AND
          (T1.CODE  >= T2.ALUM          AND
           (T1.ALUM <= T2.CODE OR T2.CODE IS NULL)) AND
          T3.TUNE    = T1.TUNE          AND
          T3.SLAM    = T1.SLAM          AND
          T3.STAR    = T1.STAR          AND
          T2.TUNE    = '240167−1447'    AND
          T2.SLAM    <= '31−dec−2002'   AND
          (T2.CODE   >= '01−jan−2002' OR T2.CODE  IS NULL) AND
         NOT EXISTS
            (SELECT T4.TUNE FROM T2 T4 WHERE
                  T4.TUNE = T2.TUNE    AND
                  T4.MAKE = '0'       AND
                  T4.SLAM = T2.SLAM    AND
                  T4.STAR    = T2.STAR AND
                  T4.SITE <> T2.SITE AND
                  T1.ALUM >= T4.ALUM AND
                  T1.CODE <= T4.CODE );
Tables:
  0 = T1
  1 = T2
  2 = T3
  3 = T2
Cross block of 4 entries
  Cross block entry 1
    Conjunct: 1.TUNE = '240167−1447'
    Conjunct: 1.SLAM <= '31−DEC−2002'
    Leaf#01 FFirst 1:T2 Card=2
      Bool: (1.CODE >= '1−JAN−2002') OR MISSING (1.CODE)
      BgrNdx1 T2_IND_1 [1:2] Fan=7
        Keys: (1.TUNE = '240167−1447') AND (1.SLAM <= '31−DEC−2002')
  Cross block entry 2
    Leaf#02 FFirst 0:T1 Card=1
```

```
      Bool: (1.TUNE = 0.TUNE) AND (1.SLAM = 0.SLAM) AND (1.STAR = 0.STAR) AND
            (0.CODE >= 1.ALUM)
      BgrNdx1 T1_IND [1:1] Fan=11
        Keys: 1.TUNE = 0.TUNE
      BgrNdx2 T1_IND_1 [3:3] Fan=8
        Keys: (1.TUNE = 0.TUNE) AND (1.SLAM = 0.SLAM) AND (1.STAR = 0.STAR)
  Cross block entry 3
    Conjunct: <agg0> = 0
    Aggregate-F1: 0:COUNT-ANY (<subselect>)
    Leaf#03 FFirst 3:T2 Card=2
      Bool: (3.TUNE = 1.TUNE) AND (3.MAKE = '0') AND (3.SLAM = 1.SLAM) AND
            (3.STAR = 1.STAR) AND (3.SITE <> 1.SITE) AND (0.ALUM >= 3.ALUM)
            AND (0.CODE <= 3.CODE)
      BgrNdx1 T2_IND_1 [3:4] Fan=7
        Keys: (3.TUNE = 1.TUNE) AND (3.SLAM = 1.SLAM) AND
              (3.STAR = 1.STAR) AND (0.ALUM >= 3.ALUM)
        Bool: (3.SLAM <= '31-DEC-2002') AND (3.TUNE = '240167-1447')
  Cross block entry 4
    Index only retrieval of relation 2:T3
      Index name  T3_IND [3:3]
        Keys: (2.TUNE = 0.TUNE) AND (2.SLAM = 0.SLAM) AND (2.STAR = 0.STAR)
 T1.CODE                    T2.ALUM                 T1.ALUM
   T2.CODE
 22-FEB-2002 00:00:00.00    1-OCT-2001 00:00:00.00   22-FEB-2002 00:00:00.00
   31-JAN-2002 00:00:00.00

 22-FEB-2002 00:00:00.00    1-FEB-2002 00:00:00.00   22-FEB-2002 00:00:00.00
   31-DEC-2003 00:00:00.00
2 rows selected
```

The predicate "T2.CODE IS NULL" is shared by two OR clauses referencing other, different tables, as in the following example.

```
   - The first predicate (T1.ALUM <= T2.CODE OR T2.CODE  IS NULL))
     references table T1 and T2, and
   - The second predicate (T2.CODE  >= '01-jan-2002' OR T2.CODE IS NULL)
     references table T2.
```

However, in the detailed strategy display, the first predicate is missing under the cross block entry 2 for table "0:T1".

As a workaround, the query works if the SQL flag 'MAX_STABILITY' is defined or the logical name RDMS$MAX_STABILITY is defined, as in the following example.

```
set flags 'max_stability';

Tables:
  0 = T1
  1 = T2
  2 = T3
  3 = T2
Cross block of 4 entries
  Cross block entry 1
    Conjunct: (1.CODE >= '1-JAN-2002') OR MISSING (1.CODE)
    Get     Retrieval by index of relation 1:T2
      Index name  T2_IND_1 [1:2]
        Keys: (1.TUNE = '240167-1447') AND (1.SLAM <= '31-DEC-2002')
  Cross block entry 2
    Conjunct: (1.SLAM = 0.ALUM) AND (1.STAR = 0.STAR) AND (0.CODE >= 1.ALUM)
```

```
            AND ((0.ALUM <= 1.CODE) OR MISSING (1.CODE))
    Get     Retrieval by index of relation 0:T1
      Index name  T1_IND [1:1]
        Keys: 1.TUNE = 0.TUNE
  Cross block entry 3
    Conjunct: <agg0> = 0
    Aggregate−F1: 0:COUNT−ANY (<subselect>)
    Conjunct: (3.MAKER = '0') AND (3.SITE <> 1.SITE) AND (0.CODE <= 3.CODE)
    Get     Retrieval by index of relation 3:T2
      Index name  T2_IND_1 [3:4]
        Keys: (3.TUNE = 1.TUNE) AND (3.SLAM = 1.SLAM) AND
              (3.STAR = 1.STAR) AND (0.ALUM >= 3.ALUM)
        Bool: (3.SLAM <= '31−DEC−2002') AND (3.TUNE = '240167−1447')
  Cross block entry 4
    Index only retrieval of relation 2:T3
      Index name  T3_IND [3:3]
        Keys: (2.TUNE = 0.TUNE) AND (2.ALUM = 0.ALUM) AND (2.STAR = 0.STAR)
 T1.CODE                   T2.ALUM                  T1.ALUM
   T2.CODE
 22−FEB−2002 00:00:00.00    1−FEB−2002 00:00:00.00   22−FEB−2002 00:00:00.00
   31−DEC−2003 00:00:00.00

1 row selected
```

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.1.10 Exit Handler in Sort Interface Removed

Previously, the internal SORT32 interface would declare an exit handler when opening an on−disk sort work file. The intent of this handler was to explicitly delete the work file in case of failure. However, when Sort was being called from within Rdb in executive mode, it was possible for the exit handler to be called during image run down and could cause an exception that caused the process to be terminated.

This problem has been corrected in Oracle Rdb Release 7.1.2.3. The exit handler is only declared when Sort is being called from user mode.

## 3.1.11 Bugchecks at PIOABW$SYNCH_PAGE + 00000564

Bug 3264272

If a process did not have sufficient quota, it was possible to encounter bugchecks like the following:

```
***** Exception at 0109556C : PIOABW$SYNCH_PAGE + 00000564
%RDMS−F−CANTWRITEDBS, error writing pages 2:367−369
−SYSTEM−F−EXQUOTA, process quota exceeded
```

While in general it is necessary to have sufficient quotas to support the total number of potential concurrent disk I/Os possible, certain operations, like asynchronous batch−writes, can safely ignore these errors. If an asynchronous batch−write encounters an EXQUOTA error, the disk write can be retried later when another attempt is made to write the buffer. The asynchronous batch−write operations have been modified to tolerate EXQUOTA errors.

Note that there are many other operations that may still fail with a bugcheck if an EXQUOTA error is encountered when attempting to read or write database disk files. For example, EXQUOTA errors encountered when attempting I/O to the recovery−unit journal (.RUJ) or after−image journal (.AIJ) will still result in a fatal bugcheck. Also, if an attempt to write out all modified buffers is unable to start any I/Os successfully then a bugcheck will still occur.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.1.12 Query Bugcheck After Logical RDMS$SET_FLAGS Set to SELECTIVITY(2)

Bug 3242615

Before the problem was fixed, a query resulted in a bugcheck when the query was executed after the sampled selectivity capability was enabled. Sampled selectivity can be enabled in several ways, one of which is to define the VMS logical name RDMS$SET_FLAGS to be SELECTIVITY(2). Below is an example of a failing query:

```
select count (*) from xxx
  where (c in (−1, 0) and b > 48955) or
        (a > 30000 and b > 49990) or
        (a > 49900 and b > 20000);
```

The problem can occur when there is an OR condition in the WHERE clause of the query.

As a workaround, if you can, enable sampled selectivity only for specific queries rather than for all queries within an application. This can be done by using the OPTIMIZE WITH SAMPLED SELECTIVITY clause on specific queries rather than defining RDMS$SET_FLAGS as SELECTIVITY(2).

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.1.13 Bugchecks or Corruption With Indexes of Type is Sorted Ranked

Bug 3009262

When updating an index of *TYPE IS SORTED RANKED*, it was possible that a bugcheck or index corruption could occur. This problem could occur if the index being updated allowed duplicates and was most likely when there existed a large number of duplicates for a particular key value and the duplicate rows were widely distributed in the database.

When a key value was changed or a row was deleted requiring the removal of a dbkey from a duplicates chain, it was possible that a bugcheck dump could be generated.

The following example shows an update that causes a key value to change. During the attempt to remove the dbkey for the affected row from the appropriate duplicates chain, Rdb generates a bugcheck dump.

```
SQL> update tt11 set f1 = 'b' where f1 = 'a' and f3 = 5 and f2 < 125;
%RDMS−I−BUGCHKDMP, generating bugcheck dump file
USER1:[BUGCHECK_DIR]RDSBUGCHK.DMP;
%COSI−F−BUGCHECK, internal consistency failure
```

The bugcheck could contain either of the two exceptions shown in the following example.

```
***** Exception at 00A32E98 : PSII2SCANGETNEXTBBCDUPLICATE + 00000120
%COSI−F−BUGCHECK, internal consistency failure

***** Exception at 00C53E4C : PSII2REMOVEDUPBBC + 0000130C
%COSI−F−BUGCHECK, internal consistency failure
```

Under rare circumstances, it was possible that the insertion of a dbkey into a duplicates chain could cause the duplicates chain to become corrupt. The corruption could be removed by dropping and recreating the offending index.

The problem can be avoided by using alternate index types or by adding columns to the index to make it more unique.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.1.14 Bugcheck at COSI_MEM_GET_VM While Doing Bitmapped Scan

Bug 3307949

Under rare circumstances, during the processing of sorted ranked indexes using bitmapped scan optimization by the dynamic optimizer, a problem in the scanning code may corrupt memory structures used within Rdb.

This memory corruption may in turn cause later access of these structures by Rdb to raise an access violation and subsequent bugcheck dump.

The bugcheck contains the following exception:

```
*SYSTEM−F−ACCVIO, access violation, reason mask=04, virtual
address=0000000079EACC30, PC=000000000027689C, PS=0000001B

***** Exception at 01190A94 : COSI_MEM_GET_VM + 00000134
```

The problem can be avoided by not enabling bitmapped scan.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.1.15 Select Count(*) on Large Table With Ranked Index Returning Wrong Results

Bug 3294338

Doing a select count(*) on a table that had a ranked index on it and had more than 2^24 rows in the table was returning wrong results. The variable being used to accumulate the count when walking the ranked tree was overflowing.

If test_tab is a table with 16850000 rows and a ranked index built on it, doing a select count(*) from test_tab will result in an answer of 16850920.

To workaround this problem, disable COUNT_SCAN optimization using RDMS$SET_FLAGS or the SET FLAGS statement with the NOCOUNT_SCAN option.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

### 3.1.16 Query With Constant Column in UNION/GROUP BY Returns Wrong Results

Bug 3336416

The following query containing a constant column in one of the UNION legs with a GROUP BY clause returns the wrong result.

```
set flags 'strategy,detail';

SELECT CTEGEN
FROM
 (SELECT CODSOC,CTEGEN,CDEV,SUM(SIGNE) AS SIGNE
    FROM (
        (SELECT CODSOC,CTEGEN,CDEV,SUM(MONTANT_SIGNE) AS SIGNE
         FROM T1
         WHERE
            CTEGEN NOT LIKE '7%' AND  PERIODE = '2001'
         GROUP BY CODSOC,CTEGEN,CDEV)
      UNION
        (SELECT CODSOC,'59100000',CDEV,SUM(MONTANT_SIGNE) AS SIGNE
         FROM T1
         WHERE
            CTEGEN LIKE '7%' AND  PERIODE = '2001'
         GROUP BY CODSOC,CTEGEN,CDEV)
         ) AS BBB
    GROUP BY CODSOC,CTEGEN,CDEV) as
    BBB(CODSOC,CTEGEN,CDEV,SIGNE)
WHERE
    SIGNE=0 AND
    CODSOC = 'MUT' AND CTEGEN LIKE '121%';
Tables:
  0 = T1
  1 = T1
Conjunct: <mapped field> = 0
Conjunct: <mapped field> LIKE '121%'
Merge of 1 entries
  Merge block entry 1
  Aggregate: 0:SUM (<mapped field>)
  Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a)
  Conjunct: '59100000' LIKE '121%'              <== See Note
  Merge of 1 entries
    Merge block entry 1
    Reduce: <mapped field>, <mapped field>, <mapped field>, <mapped field>
    Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a),
          <mapped field>(a)
    Merge of 2 entries
      Merge block entry 1
      Aggregate: 1:SUM (0.MONTANT_SIGNE)
      Conjunct: (0.CODSOC = 'MUT') AND (0.CTEGEN LIKE '121%')
      Conjunct: NOT (0.CTEGEN LIKE '7%') AND (0.PERIODE = '2001')
      Get     Retrieval by index of relation 0:T1
```

```
        Index name  T1_NDX [3:3]
          Keys: (0.PERIODE = '2001') AND (<mapped field> = 'MUT') AND (
                <mapped field> LIKE '121%')
          Bool: (0.CTEGEN LIKE '121%') AND (NOT (0.CTEGEN LIKE '7%'))
      Merge block entry 2
      Aggregate: 2:SUM (1.MONTANT_SIGNE)
      Conjunct: 1.CODSOC = 'MUT'
      Conjunct: '59100000' LIKE '121%'
      Conjunct: (1.CTEGEN LIKE '7%') AND (1.PERIODE = '2001')
      Get     Retrieval by index of relation 1:T1
        Index name  T1_NDX [3:3]
          Keys: (1.PERIODE = '2001') AND (<mapped field> = 'MUT') AND (1.CTEGEN
                 LIKE '7%')
          Bool: ('59100000' LIKE '121%') AND (1.CTEGEN LIKE '7%')
0 rows selected
```

Note:: The conjunct containing the constants in both operands is incorrectly placed outside of the merge (UNION query).

This problem occurs when the query contains the following:

1. The main table is derived from a subquery of UNION legs.
2. Each UNION leg contains a GROUP BY clause.
3. The subquery SELECT statement contains an aggregate function SUM on a column which is NOT part of the GROUP BY columns.
4. One of the UNION legs contains a constant value as the SELECT column.
5. One of the WHERE predicates of the main query references the column that is mapped to the constant value column of the UNION leg.

As a workaround, the query works if the constant column is wrapped inside a CASE statement, as in the following example.

```
SELECT CTEGEN
FROM
 (SELECT CODSOC,CTEGEN,CDEV,SUM(SIGNE) AS SIGNE
    FROM (
        (SELECT CODSOC,CTEGEN,CDEV,SUM(MONTANT_SIGNE) AS SIGNE
         FROM T1
         WHERE
            CTEGEN NOT LIKE '7%' AND  PERIODE = '2001'
         GROUP BY CODSOC,CTEGEN,CDEV)
      UNION
        (SELECT CODSOC,
!           '59100000',         ! <== wrap this constant in CASE statement
            CASE CTEGEN WHEN NULL THEN '********' ELSE '59100000' END
            CDEV,SUM(MONTANT_SIGNE) AS SIGNE
            CDEV,
          SUM(MONTANT_SIGNE) AS SIGNE
        FROM T1
        WHERE
            CTEGEN LIKE '7%' AND  PERIODE = '2001'
        GROUP BY CODSOC,CTEGEN,CDEV)
        ) AS BBB
    GROUP BY CODSOC,CTEGEN,CDEV) as
    BBB(CODSOC,CTEGEN,CDEV,SIGNE)
WHERE
    SIGNE=0 AND
    CODSOC = 'MUT' AND CTEGEN LIKE '121%';
```

```
Tables:
  0 = T1
  1 = T1
Conjunct: <mapped field> = 0
Merge of 1 entries
  Merge block entry 1
  Aggregate: 0:SUM (<mapped field>)
  Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a)
  Merge of 1 entries
    Merge block entry 1
    Reduce: <mapped field>, <mapped field>, <mapped field>, <mapped field>
    Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a),
          <mapped field>(a)
    Conjunct: 0.CTEGEN LIKE '121%'      <== See Note1
    Merge of 2 entries
      Merge block entry 1
      Aggregate: 1:SUM (0.MONTANT_SIGNE)
      Conjunct: (0.CODSOC = 'MUT') AND (0.CTEGEN LIKE '121%')
      Conjunct: NOT (0.CTEGEN LIKE '7%') AND (0.PERIODE = '2001')
      Get     Retrieval by index of relation 0:T1
        Index name  T1_NDX [3:3]
          Keys: (0.PERIODE = '2001') AND (<mapped field> = 'MUT') AND (
                <mapped field> LIKE '121%')
          Bool: (0.CTEGEN LIKE '121%') AND (NOT (0.CTEGEN LIKE '7%'))
      Merge block entry 2
      Aggregate: 2:SUM (1.MONTANT_SIGNE)
      Conjunct: 1.CODSOC = 'MUT'
      Conjunct: (1.CTEGEN LIKE '7%') AND (1.PERIODE = '2001')
      Get     Retrieval by index of relation 1:T1
        Index name  T1_NDX [3:3]
          Keys: (1.PERIODE = '2001') AND (<mapped field> = 'MUT') AND (1.CTEGEN
                 LIKE '7%')
          Bool: 1.CTEGEN LIKE '7%'
 CTEGEN
 12111090
1 row selected
```

Note1:: The correct conjunct is now generated instead of the wrong one.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.1.17 Various Bugchecks When Global Buffers in VLM Enabled

Bug 3207944

When the global buffer option "LARGE MEMORY IS ENABLED" feature was enabled, it was possible for processes to get an EXQUOTA error when disconnecting from a database. In addition, if a global checkpoint operation was in effect at the time that the database disconnect was being processed, it was possible to have the user process fail with a bugcheck dump similar to the following.

```
***** Exception at 0129BF70 : OBJMAN$UNLOCK_OBJREF + 00000070
%SYSTEM−F−ACCVIO, access violation, reason mask=00,
virtual address=0000000000000034, PC=000000000129BF70, PS=00000009
Saved PC = 012955FC : OBJMAN$BIND_OBJREF_HNDLR + 0000008C
Saved PC = 80005A68 : symbol not found
Saved PC = 800C9CCC : symbol not found
***** Exception at 012956A4 : OBJMAN$BIND_OBJREF + 00000084
Saved PC = 011D57F4 : AIJ$SUBMIT + 00000FF4
```

```
Saved PC = 012237A4 : KODBND$CKPT + 00000744
Saved PC = 012243FC : KODBND$CKPT_BLOCKING_ASTX + 000000DC
```

In addition, if the AIJ Log Server (ALS) was being utilized, it could fail with a bugcheck containing the following exception.

```
***** Exception at 001032C4 : AIJUTL$FREE_DIRTY_ARBS + 000005F4
%COSI−F−BUGCHECK, internal consistency failure
```

This problem could occur whenever an exception was encountered while disconnecting from the database and a global checkpoint operation was issued at the exact same time. When an exception occurred while disconnecting from the database, various data structures were left in an inconsistent state. If a checkpoint request was processed after the disconnect failed then the bugcheck shown above could occur.

This problem can be avoided by disabling the "LARGE MEMORY IS ENABLED" feature or by ensuring that sufficient PGFLQUO is available to accommodate the memory used for the global buffer pool.

This problem has been corrected in Oracle Rdb Release 7.1.2.3. The memory used for the global buffer pool is no longer converted to process private memory when disconnecting from the database. Also, to prevent the possibility of inconsistent data structures being left due to exceptions encountered during disconnecting from the database, if an exception occurs during disconnect the process will be deleted.

## 3.1.18 RDB$DATABASE_USERS Does Not Always Show Current Active Users

Bug 2696148

The information table RDB$DATABASE_USERS did not always contain the correct current active user information.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.1.19 Left Outer Join Query With Sub−select Overflows the Stack

Bug 3357593

A query with a left outer join with a sub−select overflows the stack.

```
SELECT
    (select ct.name from
      (select firm_id, cust_id, addr_id,
        (select name from t1  where
            t2.addr_id = t1.addr_id) as name
                 FROM t2) as ct
      where  ct.firm_id = t5.firm_id and
             ct.cust_id = t5.cust_id) as cust_name
FROM
    stock t3
    left outer join t4
        on t4.firm_id   = t3.firm_id and
           t4.trade_id  = t3.trade_id and
```

```
           t4.trade_pos = t3.trade_pos
    left outer join t5
        on t5.firm_id   = t4.firm_id and
           t5.trade_id  = t4.trade_id;
%RDB-F-IMP_EXC, facility-specific limit exceeded
-RDMS-F-XPR_STACK_OFLO, expression forces too many levels of recursion
```

The problem occurs when the query selects from three tables (t3, t4 and t5) of left outer join and contains a sub−select clause with two tables (t1 and t2) joined by the equality predicates referencing the columns from table t5.

This problem is caused by the fix made for Bug 2649215 which did not cover this particular query.

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.1.20 Signal Failing in Compound Statement

Bug 3364468

A signal in a compound statement could cause SQL/RDB to go into an infinite loop, especially if a label with a "leave label" statement was encountered. The following example shows this problem.

```
begin
declare :cnt integer = 0;
declare :tdbkey char(8);
while (exists (select c0 from update_db.module.dp$_2)) loop
        rollback;

        set transaction read write isolation level read committed
        reserving update_db.job_history for shared write;

        label_loop:

        for :dp as each row of cursor dp$_2 for
                select c0 from update_db.module.dp$_2
        do
                set :tdbkey = :dp.c0;

                delete from update_db.module.dp$_2 where current of dp$_2;

                update update_db.job_history set
                        supervisor_id = supervisor_id where dbkey = :tdbkey;

                set :cnt = :cnt + 1;
                if (:cnt >= 30) then leave label_loop;
                end if;
        end for;
        commit;
        if (:cnt >= 1) then
                signal 'JDLAY';
        end if;
end loop;
end;
```

One possible workaround would be to avoid using a "leave label" statement.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

### 3.1.21 Bugchecks in KOD$ROLLBACK and PSII2REMOVEDUPBBC With Sorted Ranked Indexes

Bug 3408974

In Oracle Rdb Release 7.1.2.2, a problem was introduced in the handling of sorted ranked indexes by the optimizer.

Depending on the structure and data distribution of index nodes in the sorted ranked index, it was possible that one of the following exceptions would be raised.

```
COSI−F−BUGCHECK, internal consistency failure
Exception occurred at KOD$ROLLBACK + 00000328

COSI−F−BUGCHECK, internal consistency failure
Exception occurred at PSII2REMOVEDUPBBC + 0000142C
```

This problem will only occur where a sorted ranked index is being used by the optimizer to filter out possible candidate dbkeys for selection.

A workaround for this problem is to use normal sorted indexes instead.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

# 3.2 SQL Errors Fixed

## 3.2.1 GET DIAGNOSTICS Keyword CONNECTION_NAME Returned Incorrect Value

Bug 880810

In previous versions of Oracle Rdb, the GET DIAGNOSTICS keyword CONNECTION_NAME did not always return the correct value. The following example shows the problem. The result 'RDB$DEFAULT_CONNECTION' was not expected but rather 'MF'.

```
SQL> declare :c char(31);
SQL>
SQL> attach 'alias aa filename sql$database';
SQL> connect to 'alias aa filename db$:mf_personnel' as 'MF';
SQL>
SQL> set connect 'MF';
SQL> begin
cont> get diagnostic :c = connection_name;
cont> end;
SQL> print :c;
 C
 RDB$DEFAULT_CONNECTION
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.1.2.3. CONNECT now correctly passes the connection name to the Rdb server so that it can be returned by GET DIAGNOSTICS.

## 3.2.2 Unexpected DATEEQLILL Error During IMPORT With CREATE INDEX or CREATE STORAGE MAP

Bug 1094071

When the SQL IMPORT statement includes CREATE STORAGE MAP or CREATE INDEX statements which use TIMESTAMP or DATE ANSI literals in the WITH LIMIT OF clauses, it fails with the following error:

```
%SQL−F−UNSDATXPR, Unsupported date expression
−SQL−F−DATEEQLILL, Operands of date/time comparison are incorrect
```

The same CREATE STORAGE MAP or CREATE INDEX statements work correctly when used outside of the IMPORT statement.

This error is generated because the SQL IMPORT statement tries to validate the data type of the column against that of the literal value. However, during this phase of the IMPORT, the table does not yet exist.

A workaround for this problem is to use DATE VMS literals in the WITH LIMIT OF clause and allow the Rdb Server to perform the data type conversion at runtime.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.2.3 SQL Did Not Correctly Handle the AS Rename or INDICATOR Clauses

Bug 1941653

The AS clause allows columns and other expressions in a SELECT expression to be renamed. The AS keyword is optional and can be ambiguous since it also could be interpreted as an INDICATOR variable.

In prior releases of Oracle RDB, the SQL syntax allowed the AS clause (with the AS keyword omitted) on almost any value expression and in some cases this syntax allowed strange and unexpected behavior.

- Missing output from the SELECT statement.
  In this example, a missing comma (,) caused the next expression to be quietly ignored. The only indication is that the E correlation name is used to rename the first expression.

  ```
  SQL> select e.last_name e.first_name from employees e limit to 1 row;
   E
   Toliver
  1 row selected
  ```

  ---

  Note

  ***Multi−segment names are legal but must be in quotes.***

  ---
- In other contexts (outside a SELECT expression), this syntax should be detected and diagnosed for the SQL programmer. In the following example, the syntax is clearly incorrect.

  ```
  SQL> select e.last_name from employees e where e.last_name e.first_name = 'Smith';
   LAST_NAME
   Smith
   Smith
  2 rows selected
  ```

This problem has been corrected in Oracle Rdb Release 7.1.2.3. Here are the new results for the preceding examples:

```
SQL> select e.last_name e.first_name from employees e limit to 1 row;
%SQL−F−INDNOTSUP, Indicator variable E.FIRST_NAME was specified in a context
where indicators are not supported
SQL>
SQL> select e.last_name from employees e where e.last_name e.first_name = 'Smith';
%SQL−F−INDNOTSUP, Indicator variable E.FIRST_NAME was specified in a context
where indicators are not supported
```

## 3.2.4 COMMIT or ROLLBACK Bugchecks in Conditional Expression

Bug 2350880

In prior versions of Oracle Rdb, attempts to COMMIT or ROLLBACK a transaction within the scope of a conditional expression (CASE, IF or WHILE) would cause a bugcheck.

- Alpha OpenVMS 7.3−1
- Oracle Rdb Server V7.0−70
- COSI−F−BUGCHECK, internal consistency failure
- Exception occurred at KOD$PREPARE + 00000228
- Called from KOD$COMMIT + 0000018C
- Called from RDMS$$INT_COMMIT_TRANSACTION + 000002BC

The following shows one example query and the resulting bugcheck dump.

```
SQL> set flags 'trace';
SQL>
SQL> begin
cont> set transaction read only;
cont>
cont> if exists
cont>      (select *
cont>       from rdb$relation_fields t1, rdb$relations t2
cont>       where t2.rdb$relation_name = t1.rdb$relation_name)
cont> then
cont>     trace 'rows exist';
cont>     commit;
cont>     set transaction read only;
cont> end if;
cont>
cont> commit;
cont> end;
~Xt: rows exist
%RDMS−I−BUGCHKDMP, generating bugcheck dump file DISK1:[TEST]RDSBUGCHK.DMP;
```

This problem has been corrected in Oracle Rdb Release 7.1.2.3. Rdb now closes the index scan from the CASE, IF, and WHILE condition prior to executing the body of the conditional expression.

## 3.2.5 Variable Not Updated After Failure in Compound Statement

Bug 2841857

In a compound statement, when the final statement which sets the SQLSTATE fails, none of the host variables set by any statement within the compound statement are updated to reflect the results of these statements. This is true regardless of whether these statements succeeded.

For example, in the following compound statement, the value of :SQL_STATE would be unchanged rather than reflecting the value set during the compound statement (02000, NO RECORD FOUND in this case). Note that this occurs even though the GET DIAGNOSTICS is actually after the failed statement since GET DIAGNOSTICS does not set the SQLSTATE.

```
SQL> attach 'filename mf_personnel';
SQL> declare :empid char(5);
SQL> declare :sql_state char(5);
SQL> ! preset the host variable values and print them
SQL> begin
cont> set :empid = '99999';
cont> set :sql_state = '99999';
cont> end;
```

```
SQL> print :sql_state;
 SQL_STATE
 99999
SQL> print :empid;
 EMPID
 99999
SQL> ! execute the compound statement to update the host variables
SQL> begin
cont>  ! this select finds no records
cont>  select employee_id into :empid from employees
cont>         where employee_id = '00100';
cont>  get diagnostics exception 1 :sql_state = RETURNED_SQLSTATE;
cont>  end;
SQL> ! :SQL_STATE should be 02000
SQL> print :sql_state;
 SQL_STATE
 99999
SQL> ! :EMPID should be unchanged because no records were found
SQL> print :empid;
 EMPID
 99999
SQL>
```

The problem has now been fixed so that host variables set by a successful statement within a compound statement are updated. In the above example, the SQL_STATE variable now displays as follows:

```
SQL> print :sql_state;
 SQL_STATE
 02000
```

As a workaround, include a "do nothing" statement as the last statement in the compound statement, which is guaranteed to set the SQLSTATE to "00000".

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.2.6 Unexpected Failure When Using ALTER ... THRESHOLDS Clause

Bug 3283407

In prior releases of Oracle Rdb, the THRESHOLDS clause for ALTER INDEX or ALTER STORAGE MAP was rejected if the logical area already existed.

The following example shows the reported error.

```
SQL> alter index u1_partitioned_table store using (attr1, attr2)
cont> in sa1 ( thresholds are (90,90,90) )  with limit of  (10,15)
cont> in sa2 ( thresholds are (90,90,90) )  with limit of  (10,25)
cont> in sa3 ( thresholds are (90,90,90) )  with limit of  (10,35);
SQL> commit;
SQL> alter index u1_partitioned_table store using (attr1, attr2)
cont> in sa1 ( thresholds are (90,90,90) )  with limit of  (10,15)
cont> in sa2 ( thresholds are (90,90,90) )  with limit of  (10,25)
cont> in sa3 ( thresholds are (90,90,90) )  with limit of  (10,35);
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-THRESHAREEXI, illegal thresholds usage - area SA1 exists,
```

```
and cannot have THRESHOLDS respecified
```

The problem was that the new threshold values were the same as those in that index or storage map partition and could have been ignored by Rdb.

This problem has been corrected in Oracle Rdb Release 7.1.2.3. Oracle Rdb now ignores the THRESHOLDS clause if it respecifies the existing thresholds for the index or storage map partition.

## 3.2.7 Unexpected Bugcheck When Oracle−style Outer Join Used With Subselect

Bug 3329186

When the Oracle style outer join modifier (+) was used in a WHERE clause that also contained a subselect with an aggregate function (SUM, AVG, MIN, MAX, COUNT, STDDEV, or VARIANCE), a bugcheck would be generated by SQL.

The following example shows a simple example of the type of query.

```
SQL> select *
cont> from employees e, salary_history sh
cont> where e.employee_id (+) = sh.employee_id
cont>   and sh.salary_amount = (select max (salary_amount)
cont>                           from salary_history);
%RDMS−I−BUGCHKDMP, generating bugcheck dump file DISK1:[DATABASE]SQLBUGCHK.DMP;
%SQL−F−BUGCHK, There has been a fatal error. Please contact your Oracle
support representative. SQL$SEMRSE − 71
```

The only workaround is to recode the query using ANSI/ISO Standard SQL syntax using the LEFT OUTER JOIN or RIGHT OUTER JOIN operators.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.2.8 Locks Not Released on Failed Compound Statement SET TRANSACTION

Bug 3168900

In prior versions of Oracle Rdb, a failed RESERVING clause of SET TRANSACTION may hold locks on other reserved tables. This problem only occurs within a compound statement (BEGIN END block). For instance, consider this example where the reserving clause for EMPLOYEES succeeds but the reserving clause on JOB_HISTORY fails.

```
SQL> begin
cont> set transaction read write
cont>     wait 1
cont>     reserving EMPLOYEES for protected write
cont>              ,JOB_HISTORY for protected write;
cont> end;
%RDB−E−LOCK_CONFLICT, request failed due to locked resource
−RDMS−F−TIMEOUT, timeout on logical area 89 "JOB_HISTORY"
−COSI−W−CANCEL, operation canceled
```

At this point, there is no transaction available to rollback. If the application were to wait or hibernate it would hold the table lock, possibly preventing access to tables.

Note

***Rdb orders the reserved tables by RDB$RELATION_ID to minimize deadlocks.***

You can workaround this problem by extracting the SET TRANSACTION statement from the compound statement or by using the LOCK TABLE statement immediately after the SET TRANSACTION statement to reserve the tables (instead of using a RESERVING clause in the SET TRANSACTION statement).

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.2.9 %SQL−F−NODBFILE, Alias Missing a Declaration With SQLMOD

Bug 1258536

This problem can arise when a SQL$MOD module is compiled with /CONNECT and has a database alias declared with a run−time resolution. In this case, if another SQL$MOD module or a Dynamic SQL module declares the same alias but with a compile−time resolution, the first call to any procedure in the module with the run−time resolution will fail with:

```
SQL−F−NODBFILE, ALIAS <alias−name> IS MISSING A DECLARATION
where <alias−name> will be the alias that fails.
```

For example, suppose SQL$MOD module1 declares an alias with run−time resolution like this:

```
declare base alias for compiletime filename 'mf_personnel'
     runtime filename db_name
```

Note: Procedures in module1 will have "db_name" as a parameter.

Suppose also that SQL$MOD module2 declares the same alias but with a compile−time resolution like this:

```
declare alias base filename 'mf_personnel'
```

Now both modules are linked with a main program that calls a procedure from module1. The call to the module1 procedure will result in:

```
SQL−F−NODBFILE, ALIAS BASE IS MISSING A DECLARATION
```

There is no workaround for this problem. There is no way to achieve run−time resolution of the alias.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.2.10 ALTER TABLE Now Allows DEFAULT Clause for AUTOMATIC Columns

Bug 3296843

In prior releases of Oracle Rdb V7.1, the DEFAULT clause was not permitted for AUTOMATIC columns that were executed for INSERT. This was because the AUTOMATIC column expression is used instead during the INSERT statement.

However, there is a need for an alternate value when a column is added to a table containing data. The ALTER TABLE statement will propagate the DEFAULT or AUTOMATIC expression to each existing row. In the case of AUTOMATIC AS or AUTOMATIC INSERT AS columns, the value set for pre−existing rows might now be acceptable based on the AUTOMATIC expression. Therefore, the DEFAULT clause is now allowed for AUTOMATIC AS and AUTOMATIC INSERT AS columns during add by ALTER TABLE ... ADD COLUMN. The default will be written to existing rows by ALTER TABLE.

The DEFAULT keyword in INSERT and UPDATE will continue to use the AUTOMATIC value expression. The DEFAULT value expression is only used by the ALTER TABLE statement.

If the DEFAULT clause specifies NULL or CAST(NULL AS ...) then no actual column update will be required as this is the implicit default applied to new columns. If all columns added by ALTER TABLE have NULL (or CAST(NULL AS ...)) as the default, then no table update will be required.

The following example shows the behavior.

```
SQL> alter table EMPLOYEES
cont>     add column LAST_UPDATED
cont>         automatic as CURRENT_TIMESTAMP
cont>         default NULL
cont> ;
SQL>
SQL> select last_name, last_updated
cont>  from EMPLOYEES
cont>  where employee_id = '00164';
 LAST_NAME        LAST_UPDATED
 Toliver          NULL
1 row selected
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.2.11 Incorrect Character Set Applied to Result of Concatenate Operator

Bug 3396134

In Oracle Rdb Release 7.1.2, an incompatibility was introduced for the concatenation operator (|| or CONCAT). The derived character set was UNSPECIFIED instead of being based on the source character strings. This particularly affected translation to a second character set (such as UNICODE, or SHIFT_JIS) that required special semantic processing based on the source character set.

The following example shows that the derived character set is UNSPECIFIED but the database literal character set is DEC_MCS.

```
>> SELECT 'a'||'b' FROM RDB$DATABASE
SQLDA [CURS_CMDLIN] Output:
 0/: CHAR(2):UNSPECIFIED::ab
```

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

# 3.3 RMU Errors Fixed

## 3.3.1 Batch Incremental Restore Allowed Updates Since the Last Full Database Restore

Bug 3140208

In the case of an incremental restore executed in batch mode where prompting was disabled, the restore was allowed if changes had occurred to the database between the last full restore and the incremental restore. This could cause database corruption since the user could not be prompted for a decision on whether to continue the restore or not. In interactive mode, the user is prompted by default with the following UPDSINFUL prompt:

```
Updates performed since the full RESTORE. Continue?
```

Now, to prevent possible database corruption for an incremental restore in the batch mode default case where /NOCONFIRM is not specified and changes have occurred to the database between the last full restore and the incremental restore (and the user cannot be prompted for a decision on whether to continue the restore), the incremental restore will not be allowed and a fatal error message will be output.

RMU will also now put out a warning message in interactive mode and the incremental restore will continue if RMU cannot prompt the user because /NOCONFIRM is specified.

To summarize the new behavior in both batch and interactive mode, if database updates since the last full restore are detected by an incremental restore:

- For interactive mode, if /CONFIRM is specified or is assumed as the default, RMU will continue to prompt to give the user a choice to continue to do the incremental restore.

  ```
  $rmu/restore/inc/nocdd/nolog/confirm bmfdb_back_inc_1
  DEVICE:[DIRECTORY]BMFDB.RDB;1, restore incrementally? [N]:Y
  Updates performed since the full RESTORE. Continue? [N]:Y
  %RMU-W-AREAEXCL, Nothing restored for storage area - AREA1_BMFDB
  %RMU-W-AREAEXCL, Nothing restored for storage area - RDB$SYSTEM
  %RMU-W-USERECCOM, Use the RMU Recover command. The journals are not available.
  ```
- For interactive mode, if /NOCONFIRM is specified, RMU will put out a warning message and continue the restore.

  ```
  $rmu/restore/inc/nocdd/nolog/noconfirm bmfdb_back_inc_1
  %RMU-W-UPDSINFULWARN, Updates have been performed since the last FULL RESTORE.
  Consider verifying your database for possible database corruption.
  %RMU-W-AREAEXCL, Nothing restored for storage area - AREA1_BMFDB
  %RMU-W-AREAEXCL, Nothing restored for storage area - RDB$SYSTEM
  %RMU-W-USERECCOM, Use the RMU Recover command. The journals are not available.
  ```
- For batch mode, if /CONFIRM is specified, RMU will continue to give an error and not allow this qualifier since there is a potential for multiple unpredictable prompts during an incremental restore.

  ```
  $rmu/restore/inc/nocdd/nolog/confirm bmfdb_back_inc_1
  %RMU-F-BATCONFIRM, confirmation not allowed in batch
  %RMU-F-FTL_RSTR, Fatal error for RESTORE operation at 19-SEP-2003 11:49:11.32
    USER      job terminated at 19-SEP-2003 11:49:11.62
  ```

- For the batch mode default case where neither /NOCONFIRM or /CONFIRM is specified, RMU will abort the incremental restore with an error message.

  ```
  $rmu/restore/inc/nocdd/nolog bmfdb_back_inc_1
  %RMU-F-UPDSINFULABRT, Updates have been performed since the last FULL RESTORE.
  Aborting the INCREMENTAL RESTORE to prevent possible database corruption.
  %RMU-F-FTL_RSTR, Fatal error for RESTORE operation at 19-SEP-2003 11:45:09.95
    USER      job terminated at 19-SEP-2003 11:45:10.30
  ```
- For batch mode where /NOCONFIRM is explicitly specified, RMU will do the same as in interactive mode: give a warning message and do the incremental restore.

  ```
  $rmu/restore/inc/nocdd/nolog/noconfirm bmfdb_back_inc_1
  %RMU-W-UPDSINFULWARN, Updates have been performed since the last FULL RESTORE.
  Consider verifying your database for possible database corruption.
  %RMU-W-AREAEXCL, Nothing restored for storage area - AREA1_BMFDB
  %RMU-W-AREAEXCL, Nothing restored for storage area - RDB$SYSTEM
  %RMU-W-USERECCOM, Use the RMU Recover command. The journals are not available.
  $exit
    USER      job terminated at 19-SEP-2003 11:51:27.17
  ```

The following example shows the old behavior. The incremental restore was allowed to continue in batch mode where the user could not be prompted.

```
$rmu/restore/inc/nocdd/nolog bmfdb_back_inc_1
%RMU-W-AREAEXCL, Nothing restored for storage area - AREA1_BMFDB
%RMU-W-AREAEXCL, Nothing restored for storage area - RDB$SYSTEM
%RMU-W-USERECCOM, Use the RMU Recover command. The journals are not available.
$exit
  USER      job terminated at 19-SEP-2003 11:55:33.62
```

The following example shows the new behavior. This is from a log of a batch incremental restore where an update has been performed since the last full restore.

```
$rmu/restore/inc/nocdd/nolog bmfdb_back_inc_1
%RMU-F-UPDSINFULABRT, Updates have been performed since the last FULL RESTORE.
Aborting the INCREMENTAL RESTORE to prevent possible database corruption.
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at 19-SEP-2003 11:45:09.95
  USER      job terminated at 19-SEP-2003 11:45:10.30
```

As a workaround, to avoid possible database corruption, do not update the database between the last full restore and the incremental restore.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.3.2 RMU/VERIFY Aborted Verification if Segmented String Problem

Bug 3200962

If there was database corruption such that RMU/VERIFY was not able to retrieve the information from the system tables necessary to do a verification of segmented strings, RMU/VERIFY aborted the verification even though it was able to continue and do the verification for other database structures besides segmented strings. Continuing the verify could have revealed the cause of the failure to retrieve the segmented string information and reveal any other database problems.

This behavior has now been changed to allow the database verification of other database structures to proceed but turn off verification of segmented strings. A new error message will be put out in this case to notify the user that segmented strings could not be verified.

```
%RMU-E-ERRRDBSEG, Error getting segmented string data from system tables.
Segmented strings will not be verified.
```

Note that this was a case where database corruption was such that RMU/VERIFY could continue the verification by ignoring segmented string verification. Depending on the nature of the database corruption, RMU/VERIFY will not always be able to continue the verification. In these cases, RMU/VERIFY will continue to put out the fatal error message.

```
%RMU-F-ABORTVER,  fatal error encountered; aborting verification
```

The following example shows the previous behavior where the verify was aborted when it could not get the information from the system tables to verify segmented strings even though the verify could have continued to verify other database structures besides segmented strings.

```
$ rmu/verify/nolog/constraints/area/index/lareas/data/snap/root-
 /segmented_strings personnel
%RDB-W-IO_ERROR, input or output error
-RDMS-F-CANTREADDBS, error reading pages 1:911-911
-RDMS-F-CHECKSUM, checksum error - computed 30F4B42D, page contained 30F3B42D
%RMU-E-ERRRDBREL, error accessing RDB$RELATIONS relation
%RMU-I-PARTLVFY,  continuing partial verification
%RMU-I-NOTREQVFY, not all requested verifications have been performed
%RMU-F-ABORTVER,  fatal error encountered; aborting verification
```

The following example shows the new behavior where the verify will not abort but continue to verify other database structures besides segmented strings.

```
$rmu/verify/constraints/area/index/lareas/data/snap/root-
/segmented_strings personnel
%RDB-W-IO_ERROR, input or output error
-RDMS-F-CANTREADDBS, error reading pages 1:911-911
-RDMS-F-CHECKSUM, checksum error - computed 30F4B42D, page contained 30F3B42D
%RMU-E-ERRRDBSEG, Error getting segmented string data from system tables.
Segmented strings will not be verified.
%RMU-E-ERRRDBREL, error accessing RDB$RELATIONS relation
%RMU-I-PARTLVFY,  continuing partial verification
%RMU-W-PAGCKSBAD, area DEVICE:[DIRECTORY]PERSONNEL.RDB;1, page
911 contains an invalid checksum
                  expected: 30F4B42D, found: 30F3B42D
%RMU-W-INVRELID, invalid relation id at dbkey 3:911:4
                  expected relation id 6, found 7
%RMU-E-CORRUPTPG, Page 911 in area
DEVICE:[DIRECTOR]PERSONNEL.RDB;1 is marked as corrupt.
%RMU-W-BADIDXREL, Index RDB$VER_REL_ID_VER_NDX either points to a non-existent
record or has multiple pointers to a record in table RDB$FIELD_VERSIONS.
                  The logical dbkey in the index is 3:911:4.
%RMU-I-NOTREQVFY, not all requested verifications have been performed
```

The workaround for this problem would be to do the RMU/VERIFY without the /SEGMENTED_STRINGS qualifier since segmented string verification is not the default.

```
$rmu/verify/constraints/area/index/lareas/data/snap/root personnel
```

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.3.3 CDD Integration Failure With a Direct RMU/CONVERT From V6.1 to V7.1

Bug 3161084

As a result of a problem in the RMU/CONVERT database conversion DIRECTLY from V6.1 to V7.1, there were two duplicate RDB$FIELD_POSITION values in the RDB$RELATION_FIELDS database system table for each of the RDB$DATABASE, RDB$INDICES, RDB$QUERY_OUTLINES and RDB$COLLATIONS system tables. This caused the failure of the database integration into the CDD with the following validation failure message.

```
%CDD−E−VALDEFFAIL,  entity
DISK:[DIRECTORY]CDD$RDB_SYSTEM_METADATA.RDB$DATABASE(1) definition
 failed validation CDD$UNIQ_DAC_SEQ_NUM
```

Note that this only happened if the conversion was directly from V6.1 to V7.1 and that the problem was with RMU/CONVERT and not the CDD which just detected the problem. This has been fixed and a direct database conversion from V6.1 to V7.1 will now integrate into the CDD without a validation failure.

The following example shows that a database, restored from a V6.1 backup file and converted directly from V6.1 to V7.1 as part of the restore, failed a CDD integration validation.

```
$ rmu/restore/cdd mfp61.rbf
%RMU−W−PREVACL, Restoring the root ACL over a pre−existing ACL.
 This is a normal condition if you are using the CDO utility.
%RMU−I−AIJRSTAVL, 0 after−image journals available for use
%RMU−I−AIJISOFF, after−image journaling has been disabled
%RMU−I−LOGCONVRT, database root converted to current structure level
%RMU−S−CVTDBSUC, database DISK:[DIRECTORY]MF_PERSONNEL.RDB;1
 successfully converted from version V6.1 to V7.1
%RMU−I−CVTCOMSUC, CONVERT committed for
 DISK:[DIRECTORY]MF_PERSONNEL.RDB;1 to version V7.1
%RMU−W−USERECCOM, Use the RMU Recover command. The journals are not available.
%CDD−I−BLRSYNINFO, unsupported entity − marked Incomplete
%CDD−I−MBLRSYNINFO, unsupported entity − marked Incomplete at mblr offset 34
%CDD−I−MBLRSYNINFO, unsupported entity − marked Incomplete at mblr offset 29
%CDD−E−VALDEFFAIL, entity
 DISK:[DIRECTORY]CDD$RDB_SYSTEM_METADATA.RDB$DATABASE(1) definition
 failed validation CDD$UNIQ_DAC_SEQ_NUM
```

One workaround to avoid this problem is to convert the V6.1 database first to V7.0 and then to V7.1 and then integrate it into the CDD. Another workaround is to create a dummy Rdb database in the CDD using PATHNAME first, before integration, in order to create a CDD$RDB_SYSTEM_METADATA instance in the CDD. This avoids the validation failure when the integrate happens.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.3.4 Aborted AIJ Backup May Cause Database Shutdown Via DBR Failure

Bug 1712408

In previous releases of Oracle Rdb, when aborting a quiet point AIJ backup of a single extensible journal, it was possible that the database would be shutdown after a DBR failure. The DBR log file, if enabled, ended with the following:

```
<timestamp> − After−image journaling is being shutdown with hard data loss
AIJ shutdown reason: journal has been possibly deleted or moved
```

When doing a quiet point backup of a single extensible journal, the journal will be truncated and re−initialized. During that phase, the file does not look like an AIJ file to Rdb. Therefore, if the backup is aborted during that phase, the DBR will not find the AIJ file and will abort, shuting down the database.

This behavior has been changed so that the DBR detects such a condition and truncates and reinitializes the AIJ file to continue the recovery. The DBR log file, if enabled, will show the following.

```
<timestamp> − Recovering AIJ information
<timestamp> − Recovering AIJ backup which failed at initialization time
<timestamp> − AIJ file has been reinitialize
<timestamp> − Initializing AIJ EOF to 1:2
```

This situation can be avoided by using multiple circular AIJs.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.3.5 RMU/CHECKPOINT Slow When Number of Nodes is One

Bug 3240378

If a database was set to have NUMBER OF CLUSTER NODES 1, and the AIJ log server (ALS) was not enabled, and there was no update activity occurring at that point in time, the ***RMU/CHECKPOINT*** command could take many minutes to complete. For example:

```
$ SQL$
  CREATE DATABASE FILENAME CKPT_TEST
    NUMBER OF CLUSTER NODES 1
    CREATE STORAGE AREA RDB$SYSTEM FILENAME CKPT_TEST;
  DISCONNECT ALL;

  ALTER DATABASE FILE CKPT_TEST
    JOURNAL IS ENABLED
      (FAST COMMIT IS ENABLED,
       −− This test only fails if there is no ALS.
       LOG SERVER IS MANUAL)
    ADD JOURNAL AIJ_1 FILENAME 'SYS$DISK:[]CKPT_TEST.AIJ';

  EXIT;
$ CREATE SUB1.COM
$ SQL$
  ATTACH 'FILENAME CKPT_TEST';
  INSERT INTO T1 VALUES (1);
  COMMIT;
  −− Wait till we are killed.
  $WAIT 1:0:0
$
```

```
$ SPAWN /NOWAIT/OUT=SUB1.LOG/PROCESS=CKPT_TIMELY_SUB @SUB1.COM
%DCL−S−SPAWNED, process CKPT_TIMELY_SUB spawned
$ WAIT 0:1:0
$ START_TIME = F$CVTIME (,,"MINUTEOFYEAR")
$ RMU/CHECKPOINT CKPT_TEST
$ END_TIME = F$CVTIME (,,"MINUTEOFYEAR")
$ CKPT_DURATION = END_TIME − START_TIME
$ IF CKPT_DURATION .GT. 2
$ THEN WRITE SYS$OUTPUT −
  "''CKPT_DURATION' minutes is too long for an RMU/CHECKPOINT command"
5 minutes is too long for an RMU/CHECKPOINT command
$ ENDIF
$ STOP CKPT_TIMELY_SUB
```

This problem can be avoided by utilizing the AIJ log server process or by setting the database number of cluster nodes greater than one.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.3.6 RMU Extract Looses NOT DETERMINISTIC Clause for Stored Functions

Bug 3280338

The RMU Extract command would loose the NOT DETERMINISTIC (also known as the VARIANT) clause for stored functions. The result was that the script created a function that defaulted to DETERMINISTIC (NOT VARIANT) which may produce unexpected results.

This problem has been corrected in Oracle Rdb Release 7.1.2.3. The NOT DETERMINISTIC clause is now displayed for SQL stored functions.

## 3.3.7 Change of Behavior When Unloading and Loading DBKEY Data Using RMU

Bug 3297161

With Release 7.1.2 of Oracle Rdb, the CREATE VIEW statement was enhanced to correctly propagate metadata from the table select expression. This included read−only attributes for columns (DBKEY, COMPUTED BY and AUTOMATIC), an indication that the column was AUTOMATIC INSERT or AUTOMATIC UPDATE, and the special DBKEY subtype of a referenced expression.

This change in behavior of CREATE VIEW requires that RMU Unload use the /VIRTUAL=COMPUTED_BY qualifier when unloading data from views (or views based on those views) that reference DBKEY data. Otherwise, this DBKEY information will not be unloaded.

Unfortunately, the special DBKEY type now saved with the data causes RMU Load to fail with the error shown in this example.

```
$ RMU/LOAD MF_PERSONNEL NEW_EMP EMP_VIEW
%RMU−F−FLDMUSMAT, Specified fields must match in number and datatype with
the unloaded data
```

This problem with RMU Load has been corrected in Oracle Rdb Release 7.1.2.3. RMU Load will no longer fail when loading DBKEY binary values into CHAR or VARCHAR columns of a table.

## 3.3.8 RMU/LOAD/DEFER_INDEX_UPDATES Accvio With a Hashed Partitioned Index

Bug 3262298

If an RMU/LOAD/DEFER_INDEX_UPDATES command was used and a partitioned hashed index was the only or primary index defined for the table being loaded, an access violation would occur and an RMUBUGCHK.DMP and RDSBUGCHK.DMP would be created. The exception in the RDSBUGCHK.DMP would occur in the routine PSII$INSERT_T. This was because the test for a hashed index did not work correctly so an attempt was made to update a hashed index using a sorted index routine. This problem has been fixed.

The following example shows that an access violation occurred if the /DEFER_INDEX_UPDATES qualifier was used for an RMU/LOAD of a table for which the only or the primary index was a partitioned hashed index.

```
SQL> att 'fi [.to]mf_personnel';
SQL> show table job_history
Information for table JOB_HISTORY

Columns for table JOB_HISTORY:
Column Name                     Data Type        Domain
-----------                     ---------        ------
EMPLOYEE_ID                     CHAR(5)          ID_NUMBER
JOB_CODE                        CHAR(4)          JOB_CODE
JOB_START                       DATE VMS         STANDARD_DATE
JOB_END                         DATE VMS         STANDARD_DATE
DEPARTMENT_CODE                 CHAR(4)          DEPARTMENT_CODE
SUPERVISOR_ID                   CHAR(5)          ID_NUMBER

Table constraints for JOB_HISTORY:
No constraints found

Constraints referencing table JOB_HISTORY:
No constraints found

Indexes on table JOB_HISTORY:
JOB_HISTORY_HASH                with column EMPLOYEE_ID
  Duplicates are allowed
  Type is Hashed Scattered
  Compression is DISABLED
Store clause:           STORE
                             using (EMPLOYEE_ID)
                                 in EMPIDS_LOW
                                     with limit of ('00200')
                                 in EMPIDS_MID
                                     with limit of ('00400')
                                 otherwise in EMPIDS_OVER
 Comment:        hash index for job_history

Storage Map for table JOB_HISTORY:
     JOB_HISTORY_MAP
```

```
Triggers on table JOB_HISTORY:
No triggers found

SQL> exit
$rmu/load/defer/log [.to]mf_personnel job_history job_history.unl
%RDMS−I−BUGCHKDMP, generating bugcheck dump file
 DEVICE:[LOGIN]RDSBUGCHK.DMP;
%SYSTEM−F−ACCVIO, access violation, reason mask=00,
 virtual address=00000050, PC=00186C66, PSL=03C00000
%RMU−I−BUGCHKDMP, generating bugcheck dump file
 DEVICE:[LOGIN]RMUBUGCHK.DMP;
%RMU−I−DATRECREAD,  274 data records read from input file.
%RMU−I−DATRECSTO,   0 data records stored.
%RMU−F−FTL_LOAD, Fatal error for LOAD operation at 13−JAN−2004 13:32:24.76
$type [login]RDSBUGCHK.DMP

***** Exception at 0093EBB0 : PSII$INSERT_T + 000000B7
```

The workaround for this problem is to not use the qualifier /DEFER_INDEX_UPDATES when doing the RMU/LOAD of a table for which the only or primary index is a partitioned hashed index.

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

## 3.3.9 RMU/VERIFY BTRBADLFT Message Severity Changed From Informational to Warning

Bug 3311940

The RMU/VERIFY BTRBADLFT message severity has been changed from informational to warning.

```
%RMU−W−BTRBADLFT, Leftmost edge of interior b−tree node at level 2 must
 have a NULL IKEY.
```

This change has been made to call attention to this message since it is recommended that the Rdb index being verified should be rebuilt if this message is received.

The following example shows that the BTRBADLFT message returned during the RMU/VERIFY of a sorted index was previously given an INFORMATIONAL severity. It will now be given a WARNING severity.

```
$RMU/VERIFY/ALL test_database.rdb

%RMU−I−BGNNDXVER, beginning verification of index TEST_INDEX
%RMU−I−BTRBADLFT, Leftmost edge of interior b−tree node at level 2 must have a
  NULL IKEY.
```

This problem has been corrected in Oracle Rdb Release 7.1.2.3.

# 3.4 LogMiner Errors Fixed

## 3.4.1 RMU /UNLOAD /AFTER_JOURNAL /CONTINUOUS Leaves Work Files Open

Previously, when using the RMU /UNLOAD /AFTER_JOURNAL /CONTINUOUS command, work files would never be closed or deleted. This could be a problem if the database had infrequent large transactions that caused the work files to grow but the used disk space would never be reclaimed.

This problem has been corrected in Oracle Rdb Release 7.1.2.3. At each AIJ switch event, work files for the RMU /UNLOAD /AFTER_JOURNAL /CONTINUOUS command that are not currently being used are closed and deleted.

## 3.4.2 RMU /UNLOAD /AFTER_JOURNAL Incorrect NULL Bit Setting When VARCHAR is Last Column

Bug 3309002

In prior versions of Oracle Rdb, the RMU /UNLOAD /AFTER_JOURNAL command could incorrectly process the null bit vector for tables with the last column being a VARCHAR data type. The LogMiner was not correctly calculating the position of the null bit vector within the data record and could pick up stray bit patterns as the null bit vector content. The effect of not using the correct null bit vector content could be NULL column values being incorrectly returned as not NULL or not NULL column values being incorrectly returned as NULL.

The following example script demonstrates one possible effect of this problem. The column Z$I6 should be returned as NULL but is being extracted as a zero value.

```
$ SQL$
  CREATE DATA FILE FOO;
  CREATE TABLE T1 (
    Z$I1 INT, Z$I2 INT, Z$I3 INT, Z$I4 INT,
    Z$I5 INT, Z$I6 INT, X$I7 INT, Z$I8 INT,
    Z$I9 INT, Z$I10 INT, Z$V1 VARCHAR(10));
  COMMIT;
  DISCONNECT ALL;
  ALTER DATA FILE FOO ADD JOURNAL J1 FILE J1 JOURNAL ENABLE;
  EXIT;
$ RMU/SET LOGMINER/ENABLE FOO.RDB
$ RMU/BACKUP/NOLOG FOO.RDB NLA0:FOO
$ RMU/BACKUP/AFTER/NOLOG FOO.RDB NLA0:FOO
$ SQL$
  ATTACH 'FILE FOO';
  INSERT INTO T1 VALUES (1,2,3,4,5,NULL,7,8,9,10,'TEST');
1 ROW INSERTED
  COMMIT;
  EXIT;
$ RMU/BACKUP/AFTER/NOLOG FOO.RDB B1.AIJ
$ RMU/UNLOAD/AFTER_JOURNAL/NOLOG FOO.RDB B1.AIJ -
  /TABLE=(NAME=T1,OUTPUT=T1.TXT) /FORMAT=DUMP
$ SEARCH T1.TXT Z$
```

```
Z$I1                            : 1
Z$I2                            : 2
Z$I3                            : 3
Z$I4                            : 4
Z$I5                            : 5
Z$I6                            : 0
Z$I8                            : 8
Z$I9                            : 9
Z$I10                           : 10
Z$V1                            : (4) TEST
$ EXIT
```

This problem has been corrected in Oracle Rdb Release 7.1.2.3. The RMU /UNLOAD /AFTER_JOURNAL command now correctly determines the location of the null bit vector within the record when the final column of the record is a VARCHAR.

### 3.4.3 RMU /UNLOAD /AFTER_JOURNAL Blank Prefix, Suffix and Separator Strings With Delimited Format Output

In prior versions of Oracle Rdb, the RMU /UNLOAD /AFTER_JOURNAL command did not allow using all combinations of blank prefix, suffix and separator strings with delimited text format output. This differed from the similar functionality in the RMU /UNLOAD command.

This problem has been corrected in Oracle Rdb Release 7.1.2.3. The RMU /UNLOAD /AFTER_JOURNAL command now allows the same combinations of blank prefix, suffix and separator strings with delimited format output as the RMU /UNLOAD command.

### 3.4.4 RMU /UNLOAD /AFTER_JOURNAL Transaction Commit Timestamp Accuracy Increase

In prior versions of Oracle Rdb, the RMU /UNLOAD /AFTER_JOURNAL command derived the transaction commit timestamp from the after−image journal "group commit" time. Multiple after−image journal records can exist within a single group and all share the timestamp. Usually, this presented no difficulty. The transaction commit time extracted by the Oracle Rdb LogMiner(tm) was "close enough" to the actual time that the transaction committed.

However, in some situations, the transaction commit time reported by LogMiner could be earlier than the transaction start time. This condition appears most often in a heavily loaded system with very short duration transactions when the AIJ Log Server (ALS) feature was enabled.

In such a configuration, it is possible for a transaction to start and complete while the ALS process is collecting information to write to the after−image journal. Because the ALS process uses a single time stamp for all groups in an I/O, the user−captured transaction start time could be later than the ALS captured group time. And because the RMU /UNLOAD /AFTER_JOURNAL command returns the transaction commit time from the "group commit" time, it may appear that the transaction committed before it started.

This situation has been changed in Oracle Rdb Release 7.1.2.3. The transaction commit time is now explicitly captured by each transaction during the commit sequence and written to the after−image journal. The RMU /UNLOAD /AFTER_JOURNAL command extracts this time value as the actual transaction commit time.

---

AIJ Content Change

***Because the LogMiner(tm) now extracts the transaction commit time from the journal in a different way, journals created with an earlier version of Rdb but whose transaction records are extracted with this release will claim an unpredictable and incorrect value for the commit time. If the accuracy of the transaction commit time is important to you, use the prior version of Oracle Rdb to unload after−image journals created with that version.***

# 3.5 Hot Standby Errors Fixed

## 3.5.1 RMU−E−LASTCMTSNINV Error on Standby Database After Journal Switch

Bug 3338635

When database replication was active, if replication was terminated immediately after a journal switchover on the master database, attempts to verify or backup the standby database could produce a LASTCMTSNINV error. For example:

```
%RMU−E−LASTCMTSNINV, Last Committed TSN (0:832) is higher than the TSN that
will be assigned next (0:640).
```

When a journal switchover occurred, the last committed TSN was being updated on the standby database but the next TSN was not being updated.

When this problem occurred, restarting replication and committing a new transaction on the master database would resynchronize the TSNs.

This problem has been corrected in Oracle Rdb Release 7.1.2.3. The next TSN is now updated when a journal switchover occurs.

# Chapter 4
# Software Errors Fixed in Oracle Rdb Release 7.1.2.2

This chapter describes software errors that are fixed by Oracle Rdb Release 7.1.2.2.

# 4.1 Software Errors Fixed That Apply to All Interfaces

## 4.1.1 Applications That Use $HIBER/$WAKE Hang in HIB

Bug 2881846

User applications that utilize the OpenVMS $HIBER/$WAKE system services to process asynchronous events could hang in "HIB" state if the database has the FAST COMMIT feature enabled. The hung process would resume executing normally if a $WAKE was issued against it by another process.

For example, the Oracle SQL/Services product utilizes $HIBER/$WAKE to coordinate events between the server processes (dispatcher and executor). The SQL/Services executor processes could sometimes hang in "HIB" state.

This problem would sometimes occur when a global checkpoint request was issued. A global checkpoint will occur whenever Oracle Rdb switches to another journal, or when a checkpoint is manually requested by issuing an RMU/CHECKPOINT command. A race condition between the database checkpoint activity and the application's usage of $WAKE could cause a $WAKE intended for the application to be consumed by the database checkpoint activity, preventing the application from properly waking from its hibernate state.

This problem can be avoided by disabling the fast commit feature. Note that this can have a significant impact on performance.

This problem has been corrected in Oracle Rdb Release 7.1.2.2.

## 4.1.2 Premature Journal Switch When Checkpoint Timer Used

Bug 3249853

When the checkpoint timeout feature (CHECKPOINT TIMED EVERY <n> SECONDS) was being utilized, it was possible for after−image journal switchovers to occur before the current journal was full. For example:

```
$ SQL$
  ALTER DATABASE FILE CKPT_TIMEOUT
    JOURNAL IS ENABLED
      (FAST COMMIT IS ENABLED
        (CHECKPOINT TIMED EVERY 5 SECONDS));
  EXIT;
$ RMU/SHOW AFTER /BACKUP_CONTEXT/OUTPUT=NL: CKPT_TIMEOUT
$ SHOW SYMBOL RDM$AIJ_CURRENT_SEQNO
  RDM$AIJ_CURRENT_SEQNO == "1"
$ SQL$
  ATTACH 'FILENAME CKPT_TIMEOUT';
  INSERT INTO T1 VALUES (1);
1 row inserted
  COMMIT;
  INSERT INTO T1 VALUES (1);
1 row inserted
  ROLLBACK;
  −− Wait for the first checkpoint timer to expire, clearing the
  −− process' current checkpoint location.
  EXIT;
```

```
$ RMU/SHOW AFTER /BACKUP_CONTEXT/OUTPUT=NL: CKPT_TIMEOUT
$
$ SHOW SYMBOL RDM$AIJ_CURRENT_SEQNO
  RDM$AIJ_CURRENT_SEQNO == "2"
```

Note in the above example how the current sequence number (RDM$AIJ_CURRENT_SEQNO) advanced to 2 after the inserting process was idle.

This problem can be avoided by disabling the CHECKPOINT TIMED EVERY <n> SECONDS feature.

This problem has been corrected in Oracle Rdb Release 7.1.2.2.

## 4.1.3 Wrong Index Retrieval is Selected in Query with Range List Predicates

Bug 3243452

The following query slows down significantly when the range list index retrieval [0:1,1:0] is incorrectly applied instead of [3:3] for greater and less than predicates:

```
set flags 'strategy,detail';
select count(*)
  from product_division pd
        inner join product_warehouse pw on
                pw.company_no = pd.company_no and pw.division_no =
                 pd.division_no and pw.product_no = pd.product_no
        left outer join wm_location wml on
                wml.company_no = pd.company_no
                and wml.division_no = pd.division_no
                and wml.warehouse_no = pw.warehouse_no
                and wml.slot_id = pw.slot_id
where pd.company_no = 1
and pw.division_no = 1
and pd.product_no not between 700000 and 799999
and pd.product_no < 900000
and pd.product_status_cd not in ('D','O','R');
Tables:
  0 = PRODUCT_DIVISION
  1 = PRODUCT_WAREHOUSE
  2 = WM_LOCATION
Aggregate: 0:COUNT (*)
Conjunct: (0.PRODUCT_NO < 700000) OR (0.PRODUCT_NO > 799999)
Cross block of 2 entries        (Left Outer Join)
  Cross block entry 1
    Cross block of 2 entries
      Cross block entry 1
        Conjunct: 0.COMPANY_NO = 1
        Conjunct: (0.PRODUCT_NO < 700000) OR (0.PRODUCT_NO > 799999)
        Conjunct: 0.PRODUCT_NO < 900000
        Conjunct: (0.PRODUCT_STATUS_CD <> 'D') AND (0.PRODUCT_STATUS_CD <> 'O')
        Conjunct: 0.PRODUCT_STATUS_CD <> 'R'
        Leaf#01 NdxOnly 0:PRODUCT_DIVISION Card=153560
          Bool: 0.DIVISION_NO = 1
          FgrNdx  PRODUCT_DIVISION_CMP_NDX [2:2] Fan=33
            Keys: (0.DIVISION_NO = 1) AND (0.COMPANY_NO = 1)
          BgrNdx1 PRODUCT_DIVISION_PRD_NDX [0:1] Fan=33
            Keys: 0.PRODUCT_NO < 900000
```

```
      Cross block entry 2
        Conjunct: 1.DIVISION_NO = 1
        Leaf#02 BgrOnly 1:PRODUCT_WAREHOUSE Card=153560
          Bool: (1.COMPANY_NO = 0.COMPANY_NO) AND (1.DIVISION_NO = 0.DIVISION_NO
                ) AND (1.PRODUCT_NO = 0.PRODUCT_NO)
          BgrNdx1 PRODUCT_WAREHOUSE_PRIMARY [0:1,1:0] Fan=33 <= Should be [3:3]
            Keys: r0: 1.PRODUCT_NO > 799999
                  r1: 1.PRODUCT_NO < 700000
            Bool: (1.PRODUCT_NO < 900000) AND (1.COMPANY_NO = 1) AND (
                  1.DIVISION_NO = 1)
  Cross block entry 2
    Conjunct: (2.COMPANY_NO = 0.COMPANY_NO) AND (2.DIVISION_NO = 0.DIVISION_NO)
              AND (2.WAREHOUSE_NO = 1.WAREHOUSE_NO) AND (2.SLOT_ID = 1.SLOT_ID)
    Index only retrieval of relation 2:WM_LOCATION
      Index name  WM_LOCATION_PRMRY [4:4]
        Keys: (2.SLOT_ID = 1.SLOT_ID) AND (2.WAREHOUSE_NO = 1.WAREHOUSE_NO) AND
              (2.DIVISION_NO = 0.DIVISION_NO) AND (2.COMPANY_NO = 0.COMPANY_NO)
           0
1 row selected
```

The query applies [3:3] index retrieval if the sql flags 'SELECTIVITY' is enabled.

The strategy output is similar to the above except for the following lines pointed to by <= in the inner Cross block entry 2.

```
      Cross block entry 2
        Conjunct: 1.DIVISION_NO = 1
        Leaf#02 BgrOnly 1:PRODUCT_WAREHOUSE Card=153560
          Bool: (1.COMPANY_NO = 0.COMPANY_NO) AND (1.DIVISION_NO = 0.DIVISION_NO
                ) AND (1.PRODUCT_NO = 0.PRODUCT_NO)
          BgrNdx1 PRODUCT_WAREHOUSE_PRIMARY [3:3] Fan=33             <=
            Keys: (1.DIVISION_NO = 0.DIVISION_NO) AND (1.PRODUCT_NO  <=
                  0.PRODUCT_NO) AND (1.COMPANY_NO = 0.COMPANY_NO)    <=
            Bool: (1.PRODUCT_NO < 900000) AND ((1.PRODUCT_NO < 700000) OR ( <=
                  1.PRODUCT_NO > 799999)) AND (1.COMPANY_NO = 1) AND (      <=
                  1.DIVISION_NO = 1)
```

This is a regression caused by the fix for Bug 2634849 in Oracle Rdb Release 7.1.2.

This problem has been corrected in Oracle Rdb Release 7.1.2.2.

## 4.1.4 Wrong Index Retrieval is Selected in Query with GTR Predicate

Bug 3144382

The wrong index retrieval is selected in a query with a GTR predicate.

```
set flags 'strategy,detail';

create index t_i on t(v1,v2,v3,v4,v5);

select * from t where
    v1='D' and v2 > 18 and
    v3 > 7 and v4 > 17 and v5 > 4
    order by v1,v2,v3,v4,v5;
```

```
Tables:
  0 = T
Conjunct: (0.V1 = 'D') AND (0.V2 > 18) AND (0.V3 > 7) AND (0.V4 > 17) AND
     (0.V5 > 4)
Get     Retrieval by index of relation 0:T
  Index name  T_I [1:1]
    Keys: 0.V1 = 'D'
    Bool: (0.V2 > 18) AND (0.V3 > 7) AND (0.V4 > 17) AND (0.V5 > 4)
...etc...
180 rows selected
```

The query takes about 10.3 seconds (elapsed time) on a VAX machine.

In Rdb Release 7.0−62, the query runs fast (approx. 1 second) with the following strategy:

```
Tables:
  0 = T
Conjunct: ((0.V1 = <cvar>) AND (0.V2 > <cvar>) AND (0.V3 > <cvar>) AND (0.V4 >
          <cvar>) AND (0.V5 > <cvar>))
Get     Retrieval by index of relation 0:T
  Index name  T_I [2:1] Bool
    Key: (0.V1 = <cvar>) AND (0.V2 > <cvar>)
    Bool: (0.V3 > <cvar>) AND (0.V4 > <cvar>) AND (0.V5 > <cvar>)
```

As a workaround, the query works if the SQL flag 'NOMAX_SOLUTION' is defined or the logical name RDMS$DISABLE_MAX_SOLUTION is defined.

```
set flags 'NOMAX_SOLUTION';

select * from t where
    v1='D' and v2 > 18 and
    v3 > 7 and v4 > 17 and v5 > 4
    order by v1,v2,v3,v4,v5;
Tables:
  0 = T
Conjunct: (0.V1 = 'D') AND (0.V2 > 18) AND (0.V3 > 7) AND (0.V4 > 17) AND
            (0.V5 > 4)
Get     Retrieval by index of relation 0:T
  Index name  T_I [2:1]
    Keys: (0.V1 = 'D') AND (0.V2 > 18)
    Bool: (0.V3 > 7) AND (0.V4 > 17) AND (0.V5 > 4)
```

The query takes about 2.22 seconds (elapsed time) on a VAX machine.

This problem has been corrected in Oracle Rdb Release 7.1.2.2.

## 4.1.5 Query Applying Zero Shortcut Returns Wrong Results

Bug 3216607

The following query, applying zero shortcut optimization, should return 7 rows.

```
set flags 'strategy,detail';
SQL> select wip_number, plan_sequence_code
cont> from pv_errors
cont> where wip_number = 'PML_257224' and plan_sequence_code = 1;
~S#0004
```

```
Leaf#01 FFirst PV_ERRORS Card=151726
  BgrNdx1 PV_ERRORS_HASH [1:1] Fan=1
  BgrNdx2 PV_ERRORS_SORTED_001 [2:2] Fan=13
~E#0004.01(1) Estim   Ndx:Lev/Seps/DBKeys 1:_6 2:1/0/0 ZeroShortcut
0 rows selected
```

A number of optimization changes have been made to the latest releases of Rdb to improve how the optimizer chooses which indexes should be used to help retrieve data.

One such improvement has been new index estimation procedures that give much more precise estimation of the effectiveness of the use of a particular index for retrieval.

This problem may occur when the optimizer incorrectly determines that one of the indices contains no records that could possibly satisfy the selection criteria and therefore immediately discontinues the processing of the query, delivering no records.

This is called 'zero shortcut' optimization which can be seen in the execution trace of the query above:

```
~E#0004.01(1) Estim   Ndx:Lev/Seps/DBKeys 1:_6 2:1/0/0 ZeroShortcut
```

As a workaround, by adding an ORDER BY clause the query returns the correct result.

```
SQL> select wip_number, plan_sequence_code
cont> from pv_errors
cont> where wip_number = 'PML_257224' and plan_sequence_code = 1
cont> order by wip_number;
 WIP_NUMBER       PLAN_SEQUENCE_CODE
  PML_257224                     1
  PML_257224                     1
  PML_257224                     1
  PML_257224                     1
  PML_257224                     1
  PML_257224                     1
  PML_257224                     1
~S#0003
Leaf#01 Sorted PV_ERRORS Card=151726
  FgrNdx  PV_ERRORS_HASH [1:1] Fan=1
  BgrNdx1 PV_ERRORS_SORTED_001 [2:2] Fan=13
 WIP_NUMBER       PLAN_SEQUENCE_CODE
 PML_257224                     1
 PML_257224                     1
 PML_257224                     1
 PML_257224                     1
 PML_257224                     1
 PML_257224                     1
~E#0003.01(1) FgrNdx  Sorted   DBKeys=8  Fetches=0+0  RecsOut=7
~E#0003.01(1) FgrNdx  Sorted   DBKeys=8  Fetches=0+0  RecsOut=7
 PML_257224                     1
7 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.1.2.2.

# 4.2 RMU Errors Fixed

## 4.2.1 Unexpected INVALID_BLR Error From RMU Load and Unload

Bug 3280604

RMU Load and RMU Unload may sometimes fail with an INVALID_BLR error in Rdb Release 7.1.2 and Release 7.1.2.1. This was reported for a database that had ROW CACHE enabled.

The offset may vary from system to system and would look similar to the following.

```
%RMU−E−OUTFILDEL, Fatal error, output file deleted
−RDB−E−INVALID_BLR, request BLR is incorrect at offset 4294967295
```

This problem has been corrected in Oracle Rdb Release 7.1.2.2.

# Chapter 5
# Software Errors Fixed in Oracle Rdb Release 7.1.2.1

This chapter describes software errors that are fixed by Oracle Rdb Release 7.1.2.1.

# 5.1 Software Errors Fixed That Apply to All Interfaces

## 5.1.1 Query With Shared Expression in Two Predicates Returns Wrong Results

Bugs 3201864 and 2285818

The following query with shared expression in two predicates should return 1 row but instead returns 2 rows.

```
set flags 'strategy,detail';

SELECT  T1.CODE,T2.ALUM, T1.ALUM, T2.CODE
  FROM T1, T2, T3 WHERE
          T2.TUNE    = T1.TUNE          AND
          T2.SLAM    = T1.SLAM          AND
          T2.STAR    = T1.STAR          AND
          (T1.CODE  >= T2.ALUM          AND
           (T1.ALUM <= T2.CODE OR T2.CODE IS NULL)) AND
          T3.TUNE    = T1.TUNE          AND
          T3.SLAM    = T1.SLAM          AND
          T3.STAR    = T1.STAR          AND
          T2.TUNE    = '240167-1447'    AND
          T2.SLAM    <= '31-dec-2002'   AND
          (T2.CODE   >= '01-jan-2002' OR T2.CODE  IS NULL) AND
         NOT EXISTS
            (SELECT T4.TUNE FROM T2 T4 WHERE
                  T4.TUNE = T2.TUNE    AND
                  T4.MAKE = '0'       AND
                  T4.SLAM = T2.SLAM    AND
                  T4.STAR    = T2.STAR AND
                  T4.SITE <> T2.SITE AND
                  T1.ALUM >= T4.ALUM AND
                  T1.CODE <= T4.CODE );
Tables:
  0 = T1
  1 = T2
  2 = T3
  3 = T2
Cross block of 4 entries
  Cross block entry 1
    Conjunct: 1.TUNE = '240167-1447'
    Conjunct: 1.SLAM <= '31-DEC-2002'
    Leaf#01 FFirst 1:T2 Card=2
      Bool: (1.CODE >= '1-JAN-2002') OR MISSING (1.CODE)
      BgrNdx1 T2_IND_1 [1:2] Fan=7
        Keys: (1.TUNE = '240167-1447') AND (1.SLAM <= '31-DEC-2002')
  Cross block entry 2
    Leaf#02 FFirst 0:T1 Card=1
      Bool: (1.TUNE = 0.TUNE) AND (1.SLAM = 0.SLAM) AND (1.STAR = 0.STAR) AND
          (0.CODE >= 1.ALUM)
      BgrNdx1 T1_IND [1:1] Fan=11
        Keys: 1.TUNE = 0.TUNE
      BgrNdx2 T1_IND_1 [3:3] Fan=8
        Keys: (1.TUNE = 0.TUNE) AND (1.SLAM = 0.SLAM) AND (1.STAR = 0.STAR)
  Cross block entry 3
    Conjunct: <agg0> = 0
```

```
    Aggregate-F1: 0:COUNT-ANY (<subselect>)
    Leaf#03 FFirst 3:T2 Card=2
      Bool: (3.TUNE = 1.TUNE) AND (3.MAKE = '0') AND (3.SLAM = 1.SLAM) AND
            (3.STAR = 1.STAR) AND (3.SITE <> 1.SITE) AND (0.ALUM >= 3.ALUM)
         AND (0.CODE <= 3.CODE)
      BgrNdx1 T2_IND_1 [3:4] Fan=7
        Keys: (3.TUNE = 1.TUNE) AND (3.SLAM = 1.SLAM) AND
              (3.STAR = 1.STAR) AND (0.ALUM >= 3.ALUM)
        Bool: (3.SLAM <= '31-DEC-2002') AND (3.TUNE = '240167-1447')
  Cross block entry 4
    Index only retrieval of relation 2:T3
      Index name  T3_IND [3:3]
        Keys: (2.TUNE = 0.TUNE) AND (2.SLAM = 0.SLAM) AND (2.STAR = 0.STAR)
 T1.CODE                    T2.ALUM                T1.ALUM
   T2.CODE
 22-FEB-2002 00:00:00.00    1-OCT-2001 00:00:00.00   22-FEB-2002 00:00:00.00
   31-JAN-2002 00:00:00.00

 22-FEB-2002 00:00:00.00    1-FEB-2002 00:00:00.00   22-FEB-2002 00:00:00.00
   31-DEC-2003 00:00:00.00
2 rows selected
```

The predicate "T2.CODE IS NULL" is shared by two OR clauses referencing other different tables, e.g.

```
   the first predicate (T1.ALUM <= T2.CODE OR T2.CODE  IS NULL)) references
table T1 and T2, and
   the second predicate (T2.CODE  >= '01-jan-2002' OR T2.CODE  IS NULL)
references table T2.
```

but, in the detailed strategy display, the first predicate is missing under the cross block entry 2 for table "0:T1".

As a workaround, the query works if the SQL flag 'MAX_STABILITY' is defined or the logical name RDMS$MAX_STABILITY is defined.

```
set flags 'max_stability';

Tables:
  0 = T1
  1 = T2
  2 = T3
  3 = T2
Cross block of 4 entries
  Cross block entry 1
    Conjunct: (1.CODE >= '1-JAN-2002') OR MISSING (1.CODE)
    Get     Retrieval by index of relation 1:T2
      Index name  T2_IND_1 [1:2]
        Keys: (1.TUNE = '240167-1447') AND (1.SLAM <= '31-DEC-2002')
  Cross block entry 2
    Conjunct: (1.SLAM = 0.ALUM) AND (1.STAR = 0.STAR) AND (0.CODE >= 1.ALUM)
              AND ((0.ALUM <= 1.CODE) OR MISSING (1.CODE))
    Get     Retrieval by index of relation 0:T1
      Index name  T1_IND [1:1]
        Keys: 1.TUNE = 0.TUNE
  Cross block entry 3
    Conjunct: <agg0> = 0
```

```
    Aggregate−F1: 0:COUNT−ANY (<subselect>)
    Conjunct: (3.MAKER = '0') AND (3.SITE <> 1.SITE) AND (0.CODE <= 3.CODE)
    Get     Retrieval by index of relation 3:T2
      Index name  T2_IND_1 [3:4]
        Keys: (3.TUNE = 1.TUNE) AND (3.SLAM = 1.SLAM) AND
              (3.STAR = 1.STAR) AND (0.ALUM >= 3.ALUM)
        Bool: (3.SLAM <= '31−DEC−2002') AND (3.TUNE = '240167−1447')
  Cross block entry 4
    Index only retrieval of relation 2:T3
      Index name  T3_IND [3:3]
        Keys: (2.TUNE = 0.TUNE) AND (2.ALUM = 0.ALUM) AND (2.STAR = 0.STAR)
 T1.CODE                    T2.ALUM                  T1.ALUM
   T2.CODE
 22−FEB−2002 00:00:00.00    1−FEB−2002 00:00:00.00   22−FEB−2002 00:00:00.00
   31−DEC−2003 00:00:00.00

1 row selected
```

This problem has been corrected in Oracle Rdb Release 7.1.2.1.

## 5.1.2 Memory Corruption When Using Explicit 2PC

Bug 3230612

It was possible for an application to experience memory corruption when the following were true:

- Explicit two−phase commit (2PC) transactions were being utilized. That is, transactions that were started and ended by using the system services SYS$START_TRANS and SYS$END_TRANS.
- The application would:
    1. Detach from the database involved in the 2PC transaction
    2. Attach to a database
    3. Start a new 2PC transaction
- The application would allocate memory after detaching from the database and before attaching to a database.

  This problem can be avoided by not using explicit two−phase commit or by remaining attached to all databases.

  For more information regarding explicit two−phase commit transactions see the Guide to Distributed Transactions.

  This problem has been corrected in Oracle Rdb Release 7.1.2.1.

## 5.1.3 DBR Bugchecks at DBR$VALIDATE_OPT_PAGES + 000001C4

When the PAGE TRANSFER VIA MEMORY was enabled, it was possible for Database Recovery Processes (DBRs) to fail with the following exception:

```
***** Exception at 00074284 : DBR$VALIDATE_OPT_PAGES + 000001C4
%SYSTEM−F−ACCVIO, access violation, reason mask=00,
virtual address=0000000000000000, PC=0000000000074284, PS=0000001B
```

This bugcheck could occur when an AIJ Backup Server (ABS) or AIJ Log Server (ALS) process would fail. The DBR process would attempt to recover global buffer data structures for those processes even though the processes did not have any global buffers allocated. This problem was introduced in Release 7.1.2.

This problem can be avoided by using PAGE TRANSFER VIA DISK or by disabling the ALS and ABS processes.

This problem has been corrected in Oracle Rdb Release 7.1.2.1.

# 5.2 Row Cache Errors Fixed

## 5.2.1 RMU Online Backup Can Fail When Using the "Snapshots in Row Cache" feature

Bug 3214111

When using the "Snapshots in Row Cache" feature with Oracle Rdb Release 7.1.2, it was possible for an online RMU backup command to fail with a bugcheck with an exception similar to "RMU−F−CANTREADDBS, error reading pages 151:4294967295− 4294967295".

This problem was caused by the RCS process not correctly writing snapshot records from cache back to database snapshot storage areas at the start of the backup operation.

This problem has been corrected in Oracle Rdb Release 7.1.2.1. The RCS process now always writes all required snapshot records to the database from the row cache.

# Chapter 6
# Software Errors Fixed in Oracle Rdb Release 7.1.2

This chapter describes software errors that are fixed by Oracle Rdb Release 7.1.2.

# 6.1 Software Errors Fixed That Apply to All Interfaces

## 6.1.1 Area Access Fails After Online Storage Area Creation

Bug 3120908

In prior versions of Oracle Rdb Release 7.1, it was possible for access to storage areas created online to fail if the database was open cluster−wide. This problem was caused by an incorrect reference to the name of the newly added snapshot storage area.

The following sequence of events demonstrates one possible failure case using "NODEA" and "NODEB".

```
NODEA:  $ SQL
        SQL> CREATE DATABASE FILENAME DB
              RESERVE 10 STORAGE AREAS
              CREATE STORAGE AREA AREA1
              CREATE STORAGE AREA AREA2;
        SQL> CREATE TABLE T1 (COL1 INTEGER);
        SQL> COMMIT;
        SQL> EXIT;
        $ RMU /OPEN /WAIT DB

NODEB:  $ RMU /OPEN /WAIT DB

NODEA:  $ SQL
        SQL> ALTER DATABASE FILENAME DB ADD STORAGE AREA AREA3;
        SQL> −− Do not exit from SQL

NODEB:  $ SQL$
        SQL> ATTACH 'FILE DB';
        SQL> CREATE INDEX IDX_T1 ON T1 (COL1) STORE IN AREA3;
        %RDMS−I−BUGCHKDMP, generating bugcheck dump file
        DGA0:[DB]RDSBUGCHK.DMP;
        %RDMS−I−BUGCHKDMP, generating bugcheck dump file
        DGA0:[DB]SQLBUGCHK.DMP;
        %SYSTEM−F−ACCVIO, access violation, reason mask=00, virtual
        address=0000000000000000, PC=00000000003AD32C, PS=0000001B
```

The bugcheck "footprint" for this particular case for Oracle Rdb Release 7.1.1.0.1 is:

```
Exception occurred at PIO$READY + 000010B8
SYSTEM−F−ACCVIO, access violation
Called from PIO$READY + 00001964
Called from RDMS$$KOD_CREATE_LAREA + 00000324
Called from RDMS$$CREATE_INDEX_INFO + 00002904
```

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.2 Page Transfer Via Memory Feature Now Available

The global buffer feature PAGE TRANSFER VIA MEMORY has been re−introduced in Release 7.1.2 of Oracle Rdb. This feature was disabled in a previous release of Oracle Rdb due to occasional problems with database corruption when the feature was enabled. The corruption problems have been resolved and the feature is again available for use.

The PAGE TRANSFER VIA MEMORY feature may provide considerable performance improvements in application environments where there is a lot of contention for database pages. To determine if this is a significant factor in the performance of a database use the ***RMU/SHOW STATISTICS*** utility, go to the "PIO Statistics−−Data Writes" screen, and examine the "blocking AST" counter. If that number represents a significant percentage of the overall number of buffer writes (the "unmark buffer" counter) then it may be worthwhile to try the PAGE TRANSFER VIA MEMORY feature to see if it improves performance. Note that this feature can only be used if global buffers are enabled, after−image journaling is enabled, fast commit is enabled, and either the number of cluster nodes is set to one, or the "SINGLE INSTANCE" clause was specified for the number of cluster nodes. See the *Guide to Database Performance and Tuning* for more information regarding this feature.

## 6.1.3 UNION Query With Two Left Outer Joins in First Leg Returns Wrong Results

Bugs 3076004 and 2529598

The following UNION query with left outer join should return 1 row.

```
set flags 'strategy,detail';
select routing_id from
     (select
     C4.ROUTING_ID,
 C2.FY,
 C2.PAY_PERIOD
          from
       ROSTER as C2
  left outer join
      WORK_AUTH as C4
   on (C2.AUTH_NBR = C4.AUTH_NBR)
       left outer join
           TAX_BEN as C5
        ON (C2.AUTH_NBR = C5.TAX_BEN_NBR)
    union
      select
 C6.ROUTING_ID,
 C7.FY,
 C7.PAY_PERIOD
          from
       WORK_AUTH as C6, PAY_PERIOD as C7)
       AS DT (ROUTING_ID, FY, PAY_PERIOD)
    where
    fy='2004' and pay_period='01' and routing_id = 'R91297';
Tables:
  0 = ROSTER
  1 = WORK_AUTH
  2 = TAX_BEN
```

```
  3 = WORK_AUTH
  4 = PAY_PERIOD
Merge of 1 entries
  Merge block entry 1
  Reduce: <mapped field>, <mapped field>, <mapped field>
  Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a)
  Merge of 2 entries
    Merge block entry 1
    Cross block of 2 entries (Left Outer Join)
      Cross block entry 1
        Cross block of 2 entries (Left Outer Join)
          Cross block entry 1
            Leaf#01 BgrOnly 0:ROSTER Card=2
              BgrNdx1 ROSTER_NDX [2:2] Fan=15
                Keys: (<mapped field> = '2004') AND (<mapped field> = '01')
          Cross block entry 2
            Leaf#02 BgrOnly 1:WORK_AUTH Card=1
              Bool: 0.AUTH_NBR = 1.AUTH_NBR
              BgrNdx1 WORK_AUTH_NDX [1:1] Fan=16
                Keys: <mapped field> = 'R91297'
      Cross block entry 2
        Conjunct: 0.AUTH_NBR = 2.TAX_BEN_NBR
        Index only retrieval of relation 2:TAX_BEN
          Index name  TAX_BEN_NDX [1:1]
            Keys: 0.AUTH_NBR = 2.TAX_BEN_NBR
    Merge block entry 2
    Cross block of 2 entries
      Cross block entry 1
        Conjunct: 3.ROUTING_ID = 'R91297'
        Index only retrieval of relation 3:WORK_AUTH
          Index name  WORK_AUTH_NDX [1:1]
            Keys: <mapped field> = 'R91297'
      Cross block entry 2
        Conjunct: (4.FY = '2004') AND (4.PAY_PERIOD = '01')
        Index only retrieval of relation 4:PAY_PERIOD
          Index name  PAY_PERIOD_NDX [2:2]
            Keys: (<mapped field> = '2004') AND (<mapped field> = '01')
 ROUTING_ID
 R91297
 NULL
2 rows selected
```

This problem is similar to the previous Bug 2529598 where the fix did not cover the current case with two left outer joins in the first UNION leg. The conjunct "routing_id = 'R91297'" is generated at the top of the second UNION (merge) leg but not at the top of the first leg and thus returns the wrong result.

The query works if the legs of the UNION clause are swapped, as in the following example:

```
select routing_id from
     (
      select
 C6.ROUTING_ID,
 C7.FY,
 C7.PAY_PERIOD
         from
       WORK_AUTH as C6, PAY_PERIOD as C7
    union
      select
      C4.ROUTING_ID,
```

```
 C2.FY,
 C2.PAY_PERIOD
          from
       ROSTER as C2
  left outer join
      WORK_AUTH as C4
   on (C2.AUTH_NBR = C4.AUTH_NBR)
       left outer join
           TAX_BEN as C5
        ON (C2.AUTH_NBR = C5.TAX_BEN_NBR)
        )
        AS DT (ROUTING_ID, FY, PAY_PERIOD)
    where
    fy='2004' and pay_period='01' and routing_id = 'R91297';
Tables:
  0 = WORK_AUTH
  1 = PAY_PERIOD
  2 = ROSTER
  3 = WORK_AUTH
  4 = TAX_BEN
Conjunct: <mapped field> = 'R91297'
Merge of 1 entries
  Merge block entry 1
  Reduce: <mapped field>, <mapped field>, <mapped field>
  Sort: <mapped field>(a), <mapped field>(a), <mapped field>(a)
  Conjunct: 1.FY = '2004'
  Conjunct: 1.PAY_PERIOD = '01'
  Merge of 2 entries
    Merge block entry 1
    Cross block of 2 entries
      Cross block entry 1
        Conjunct: 0.ROUTING_ID = 'R91297'
        Index only retrieval of relation 0:WORK_AUTH
          Index name  WORK_AUTH_NDX [1:1]
            Keys: <mapped field> = 'R91297'
      Cross block entry 2
        Conjunct: (1.FY = '2004') AND (1.PAY_PERIOD = '01')
        Index only retrieval of relation 1:PAY_PERIOD
          Index name  PAY_PERIOD_NDX [2:2]
            Keys: (<mapped field> = '2004') AND (<mapped field> = '01')
    Merge block entry 2
    Cross block of 2 entries        (Left Outer Join)
      Cross block entry 1
        Cross block of 2 entries        (Left Outer Join)
          Cross block entry 1
            Leaf#01 BgrOnly 2:ROSTER Card=2
              BgrNdx1 ROSTER_NDX [2:2] Fan=15
                Keys: (<mapped field> = '2004') AND (<mapped field> = '01')
          Cross block entry 2
            Conjunct: 2.AUTH_NBR = 3.AUTH_NBR
            Get     Retrieval by index of relation 3:WORK_AUTH
              Index name  WORK_AUTH_NDX [0:0]
      Cross block entry 2
        Leaf#02 BgrOnly 4:TAX_BEN Card=1
          Bool: 2.AUTH_NBR = 4.TAX_BEN_NBR
          BgrNdx1 TAX_BEN_NDX [1:1] Fan=17
            Keys: 2.AUTH_NBR = 4.TAX_BEN_NBR
 ROUTING_ID
 R91297
1 row selected
```

Notice that the "Conjunct: <mapped field> = 'R91297'" is now generated on top of both UNION legs. Even though the conjunct is NOT efficiently optimized by being pushed down to the second leg at least the query works correctly.

There is no known workaround other than the above−mentioned query with UNION legs swapped.

This problem has been corrected in Oracle Rdb Release 7.1.2.

### 6.1.4 Error %RDMS−E−NOSOL_FOUND in Full Outer Join Query

Bug 2669656

The following full outer natural join query between columns of different data types fails using the cross strategy (it should succeed applying the match strategy):

```
create table x1 (f1 char(10));
create table x2 (f1 integer);
create table x3 (f1 char(10));

insert into x1 value ('1');
insert into x2 value (1);
insert into x2 value (−1);
insert into x3 value ('1');
insert into x3 value ('−1');

select * From x1 natural full outer join x2;
~S: Full OJ query with cross strategy was not possible
%RDMS−E−NOSOL_FOUND, No possible solution has been found by Rdb optimizer
```

As a workaround, the following full outer join between columns of the same data type works applying a match strategy.

```
select * From x1 natural full outer join x3;
Tables:
  0 = X1
  1 = X3
Match  (Full Outer Join)
  Outer loop
    Sort: 0.F1(a)
    Get Retrieval sequentially of relation 0:X1
  Inner loop
    Temporary relation
    Sort: 1.F1(a)
    Get Retrieval sequentially of relation 1:X3
 F1
 −1
 1
2 rows selected
```

The following query should apply a match strategy as suggested by the outline bug_outline.

```
create outline bug_outline
id '6CBC3F110B75FD48C256CE94DCEB8A1F'
mode 0
```

```
as (
  query (
-- For loop
    subquery (
      X1 0    access path sequential
        join by match to
      X2 1    access path sequential
      )
    )
  )
compliance optional     ;

select * from x1 , x2 where x1.f1 = x2.f1;
~S: Outline "BUG_OUTLINE" used
~S: Full compliance with the outline was not possible
Tables:
  0 = X1
  1 = X2
Cross block of 2 entries
  Cross block entry 1
    Get     Retrieval sequentially of relation 0:X1
  Cross block entry 2
    Conjunct: 0.F1 = 1.F1
    Get     Retrieval sequentially of relation 1:X2
 X1.F1              X2.F1
 1                      1
1 row selected
```

But the query works if one of the join predicates is cast as the same data type as the other as in the following example.

```
create outline bug_good_outline
id '0EEB4BC11CF012EDB10AEA1C16EC0E70'
mode 0
as (
  query (
-- For loop
    subquery (
      X1 0    access path sequential
        join by match to
      X2 1    access path sequential
      )
    )
  )
compliance optional     ;

select * from x1 , x2 where cast(x1.f1 as integer) = x2.f1;
~S: Outline "BUG_GOOD_OUTLINE" used
Tables:
  0 = X1
  1 = X2
Conjunct: CAST (0.F1 AS INT) = 1.F1
Match
  Outer loop
    Sort: CAST (0.F1 AS INT)(a)
    Get Retrieval sequentially of relation 0:X1
  Inner loop
    Temporary relation
    Sort: 1.F1(a)
    Get Retrieval sequentially of relation 1:X2
```

```
 X1.F1              X2.F1
 1                      1
1 row selected
```

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.5 Query With EXISTS Clause and COMPUTED BY Column Returns Wrong Results

Bug 2452636

The following query, checking for EXISTS clause with an equality predicate involving a column of "COMPUTED BY", returns wrong results when applying a match strategy.

```
Columns for table T2:
Column Name     Data Type  Domain
-----------     ---------  ------
A_DATE                          DATE VMS
A_SEC_NAME                      CHAR(12)
A_DET_METHOD                    INTEGER
A_CALC_PRICE                    INTEGER
A_BAD_CMPBY                     INTEGER
 Computed:   by
      case
    when (A_CALC_PRICE > 0 and
         A_CALC_PRICE =
   (select T0.A_PRICE from T0
        where T0.A_SEC_NAME = T0.A_SEC_NAME
    and T0.A_DATE = T0.A_DATE
    and T0.A_PRICE_SOURCE = 'GIC'
    and T0.A_PRICE_TYPE = 'STL' limit to 1 rows))
          then A_DET_METHOD
       else 5
          end

Indexes on table T2:
T2_NDX                          with column A_SEC_NAME
  and column A_DATE

set flags 'strategy,detail';
sel a_data from T1 s
where exists (select * from T2 p
                where p.A_SEC_NAME=s.A_SEC_NAME and
                      p.a_date=s.a_date and
                      p.a_bad_cmpby =   2);
Tables:
  0 = T1
  1 = T2
  2 = T0
Conjunct: <agg0> <> 0
Match
  Outer loop
    Sort: 0.A_SEC_NAME(a), 0.A_DATE(a)
    Cross block of 2 entries
      Cross block entry 1
        Aggregate: 1:VIA (2.A_PRICE)
        Firstn: 1
        Leaf#01 FFirst 2:T0 Card=10
```

```
          Bool: (2.A_SEC_NAME = 2.A_SEC_NAME) AND (2.A_DATE = 2.A_DATE) AND (
                2.A_PRICE_SOURCE = 'GIC') AND (2.A_PRICE_TYPE = 'STL')
          BgrNdx1 T0_NDX [2:2] Fan=9
            Keys: (2.A_PRICE_SOURCE = 'GIC') AND (2.A_PRICE_TYPE = 'STL')
            Bool: (2.A_SEC_NAME = 2.A_SEC_NAME) AND (2.A_DATE = 2.A_DATE)
      Cross block entry 2
        Leaf#02 BgrOnly 0:T1 Card=2
          BgrNdx1 T1_NDX [0:0] Fan=10
  Inner loop      (zig-zag)
    Aggregate-F1: 0:COUNT-ANY (<subselect>)
    Get     Retrieval by index of relation 1:T2
      Index name  T2_NDX [0:0]
      A_DATA
          14
          10
2 rows selected
```

Notice that the following equality predicate involving the "Computed by" column A_BAD_CMPBY is missing in the above strategy, and thus, the query returns the wrong result of 2 rows.

```
       "p.a_bad_cmpby =   2"

   where p.a_bad_cmpby represents the following CASE statement:

   case
        when (A_CALC_PRICE > 0 and
      A_CALC_PRICE =
       (select T0.A_PRICE from T0
     where T0.A_SEC_NAME = T0.A_SEC_NAME
      and T0.A_DATE = T0.A_DATE
      and T0.A_PRICE_SOURCE = 'GIC'
      and T0.A_PRICE_TYPE = 'STL' limit to 1 rows))
  then A_DET_METHOD
   else 5
     end
```

Notice that the select statement references only a single table T0.

The missing conjunct is supposed to be generated in the following format with the aggregate value represented by <agg0>.

```
Conjunct: CASE (WHEN ((1.A_CALC_PRICE > 0) AND (1.A_CALC_PRICE = <agg0>))
                THEN 1.A_DET_METHOD ELSE 5) = 2
```

The reason that the conjunct is not created in the query is that the inner match leg contains only the context "1:T2" while the conjunct involves an aggregate value with external context "0:T0" from the outer match leg.

Here is one workaround for the problem. This query works if an outline is used to change the strategy from match to cross, since the inner cross leg contains both the contexts from the outer and inner cross legs.

```
sel a_data from T1 s
where exists (select * from T2 p
                where p.A_SEC_NAME=s.A_SEC_NAME and
                      p.a_date=s.a_date and
                      p.a_bad_cmpby =   2);
```

```
~S: Outline "TEST_BUG_OUTLINE" used
Tables:
  0 = T1
  1 = T2
  2 = T0
Cross block of 3 entries
  Cross block entry 1
    Aggregate: 0:VIA (2.A_PRICE)
    Firstn: 1
    Leaf#01 FFirst 2:T0 Card=10
      Bool: (2.A_SEC_NAME = 2.A_SEC_NAME) AND (2.A_DATE = 2.A_DATE) AND (
            2.A_PRICE_SOURCE = 'GIC') AND (2.A_PRICE_TYPE = 'STL')
      BgrNdx1 T0_NDX [2:2] Fan=9
        Keys: (2.A_PRICE_SOURCE = 'GIC') AND (2.A_PRICE_TYPE = 'STL')
        Bool: (2.A_SEC_NAME = 2.A_SEC_NAME) AND (2.A_DATE = 2.A_DATE)
  Cross block entry 2
    Leaf#02 FFirst 0:T1 Card=2
      BgrNdx1 T1_NDX [0:0] Fan=10
  Cross block entry 3
    Conjunct: <agg1> <> 0
    Aggregate-F1: 1:COUNT-ANY (<subselect>)
    Leaf#03 FFirst 1:T2 Card=2
      Bool: (1.A_SEC_NAME = 0.A_SEC_NAME) AND (1.A_DATE = 0.A_DATE) AND (CASE (
            WHEN ((1.A_CALC_PRICE > 0) AND (1.A_CALC_PRICE = <agg0>)) THEN
            1.A_DET_METHOD ELSE 5) = 2)
      BgrNdx1 T2_NDX [2:2] Fan=10
        Keys: (1.A_SEC_NAME = 0.A_SEC_NAME) AND (1.A_DATE = 0.A_DATE)
-- Rdb Generated Outline : 25-JUN-2003 11:30
create outline QO_F5E5D311487F5E17_00000000
id 'F5E5D311487F5E1776847CFB5A9C308B'
mode 0
as (
  query (
-- For loop
    subquery (
      subquery (
        T0 2    access path index       T0_NDX
        )
        join by cross to
      T1 0    access path index       T1_NDX
        join by cross to
      subquery (
        T2 1    access path index       T2_NDX
        )
      )
    )
  )
compliance optional     ;
0 rows selected
```

Here is another possible workaround. The query also works if the COMPUTED BY column is changed to join the tables T0 and T2 instead of single table T0.

```
alter table T2 add column A_GOOD_CMPBY
    computed by
     case
      when (A_CALC_PRICE > 0 and
          A_CALC_PRICE =
     (select T0.A_PRICE from T0
   where T0.A_SEC_NAME = T2.A_SEC_NAME
```

```
         and T0.A_DATE = T2.A_DATE
         and T0.A_PRICE_SOURCE = 'GIC'
         and T0.A_PRICE_TYPE = 'STL' limit to 1 rows))
        then A_DET_METHOD
        else 5
    end;

sel a_data from T1 s
where exists (select * from T2 p
                where p.A_SEC_NAME=s.A_SEC_NAME and
                      p.a_date=s.a_date and
                      p.a_good_cmpby =   2);
Tables:
  0 = T1
  1 = T2
  2 = T0
Conjunct: <agg0> <> 0
Match
  Outer loop      (zig-zag)
    Get     Retrieval by index of relation 0:T1
      Index name  T1_NDX [0:0]
  Inner loop
    Aggregate: 0:COUNT-ANY (<subselect>)
    Cross block of 2 entries
      Cross block entry 1
        Get     Retrieval by index of relation 1:T2
          Index name  T2_NDX [0:0]
      Cross block entry 2
        Conjunct: CASE (WHEN ((1.A_CALC_PRICE > 0) AND (1.A_CALC_PRICE = <agg1>)
                  ) THEN 1.A_DET_METHOD ELSE 5) = 2
        Aggregate: 1:VIA (2.A_PRICE)
        Firstn: 1
        Leaf#01 FFirst 2:T0 Card=10
          Bool: (2.A_SEC_NAME = 1.A_SEC_NAME) AND (2.A_DATE = 1.A_DATE) AND (
                2.A_PRICE_SOURCE = 'GIC') AND (2.A_PRICE_TYPE = 'STL')
          BgrNdx1 T0_NDX [4:4] Fan=9
            Keys: (2.A_PRICE_SOURCE = 'GIC') AND (2.A_PRICE_TYPE = 'STL') AND (
                  2.A_SEC_NAME = 1.A_SEC_NAME) AND (2.A_DATE = 1.A_DATE)
0 rows selected
```

Notice that the conjunct with aggregate value now appears in the Cross block entry 2 under the Inner loop of the match strategy.

This query works since the context "1:T2" is joined by cross strategy with the context "2:T0" and the conjunct with aggregate value is properly resolved with all the contexts available.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.6 Bugcheck in DIO$FREE_CURRENT_LOCK for Sorted Ranked Indexes

Bug 2874671

If a cursor was used to fetch rows, and the transaction was committed between fetches, and the retrieval strategy involved a backwards scan of an index of *TYPE IS SORTED RANKED*, then Rdb could generate a bugcheck dump.

The problem occurred because Oracle Rdb mistakenly released a lock prematurely. Later, when Oracle Rdb was truly finished with the resource, a bugcheck occurred because a second attempt was made to release the same lock.

The following example uses the MF_PERSONNEL database to demonstrate the problem.

```
SQL> at 'f mf_personnel';
SQL> drop index EMP_EMPLOYEE_ID;
SQL> commit;
SQL> create unique index EMP_EMPLOYEE_ID
cont>     on EMPLOYEES (EMPLOYEE_ID asc)
cont>     type is SORTED ranked
cont>     node size 430
cont>     disable compression;
SQL> commit work;
SQL> declare transaction read write isolation level read committed;
SQL> declare t2 table cursor with hold preserve all for
cont> select    *
cont> from      employees
cont> where     employee_id > '00300'
cont> and       employee_id < '00400'
cont> order by employee_id desc;
SQL> open t2;
SQL> fetch t2;
 EMPLOYEE_ID   LAST_NAME        FIRST_NAME   MIDDLE_INITIAL
   ADDRESS_DATA_1              ADDRESS_DATA_2         CITY
      STATE   POSTAL_CODE   SEX    BIRTHDAY      STATUS_CODE
 00374         Andriola         Leslie       Q
   111 Boston Post Rd.                                Salisbury
      NH      03268         M      19-Mar-1955   1

SQL> commit;
SQL> fetch t2;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file MBRADLEY_USR:[BRADLEY]RDSBUGCHK.DMP;
```

The problem can be avoided by disabling the backward scan feature using the *RDMS$DISABLE_REVERSE_SCAN* logical name.

The problem does not occur if all rows are fetched in the same transaction.

The problem does not occur if the index being scanned is not of *TYPE IS SORTED RANKED*.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.7 Illegal Page Count Error in the Dynamic Optimizer

Bug 3045841

When a very large table was queried, and the access strategy used the dynamic optimizer, and the first two indexes used both returned more than 1024 dbkeys, an illegal page count error could be generated.

To encounter this error, the table had to have close to one billion (1,000,000,000) rows.

In addition, the retrieval strategy for the query must use the dynamic optimizer where at least two indices return more than 1024 dbkeys.

In this case, the dynamic optimizer will allocate memory for a dbkey bitmap. In calculating the size of that bitmap, integer (signed longword) arithmetic was used, and an integer overflow could cause the calculation to result in a negative number being used as the requested amount of memory.

The following example shows a query on a table containing one billion (1,000,000,000) rows.

```
SQL> select * from t1
cont>          where f1 >0 and f2 > 0
cont>          optimize for total time;
%COSI−F−UNEXPERR, unexpected system error
−SYSTEM−F−ILLPAGCNT, illegal page count parameter
```

A bugcheck dump may also be generated with the following exception and call sequence:

```
***** Exception at 00FE3664 : COSI_MEM_GET_VM + 000007B4
%COSI−F−UNEXPERR, unexpected system error
−SYSTEM−F−ILLPAGCNT, illegal page count parameter
Saved PC = 00FE2E68 : COSI_MEM_GET_POOL + 00000048
Saved PC = 00E3CCE8 : RDMS$$EXE_LEAF + 00001A38
Saved PC = 00E290B4 : RDMS$$EXE_OPEN + 00000764
```

The problem would not occur if any of the conditions described above were not true.

The problem can be avoided by disabling the dynamic optimizer for the query by using a query outline with the clause *EXECUTION OPTIONS NONE*.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.8 Ranked Index Overflow Node Corruption on Insert

Bug 3009262

A problem in the way Rdb chooses the insertion point for the dbkey in a Ranked Index duplicate node may, in rare cirumstances, cause a corruption of the index entry overflow node. This node corruption may manifest itself as a bugcheck when trying to access the records associated with the overflow node, as in the following example.

```
***** Exception at 00C72D78 : PSII2SCANGETNEXTBBCDUPLICATE + 00000128
%COSI−F−BUGCHECK, internal consistency failure
```

Or, this node corruption may be seen as a corrupt index warning when RMU/VERIFY/INDEX is used to verify the ranked index. For example:

```
%RMU−I−BTRDUPCAR, Inconsistent duplicate cardinality (C1) of 221 specified
                  for entry 1 at dbkey 85:807:1.
                  Actual count of duplicates is 251.
%RMU−I−BTRERPATH, parent B−tree node of 85:807:1 is at 85:806:0
%RMU−I−BTRROODBK, root dbkey of B−tree is 85:806:0
%RMU−W−DATNOTIDX, Row in table TT11 is not in any indexes.
                  Logical dbkey is 83:82:2.
%RMU−W−BADIDXREL, Index INDTT either points to a non−existent record or
```

```
                  has multiple pointers to a record in table TT11.
                  The logical dbkey in the index is 83:127:8.
```

A dump of the offending overflow node will show that even though the reference dbkey is correct, all the dbkeys in the overflow node may be incorrect and possibly not valid record dbkeys.

The problem may only occur on Sorted Ranked index entries that are volatile enough so that more than one overflow node is required to hold the duplicates and that, due to removal of dbkeys from the entry, at least one non−final overflow node has subsequently been removed from the entry.

A subsequent insertion of a new duplicate in that entry may, if the insertion dbkey happens to be greater than the last dbkey in an overflow node but less than the reference dbkey of the next overflow node, incorrectly update the overflow node causing the subsequent corruption.

This problem only occurs with Sorted Ranked indexes.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.9 Incorrect Results From a Reverse Scan on a Ranked Index

Bug 2874671

When an index of *TYPE IS SORTED RANKED* was used for retrieval and the scan was a reverse scan, the wrong number of rows may have been returned.

The problem and symptoms are similar to that described in Section 6.1.43, however this problem is separate and relates to reverse index scans on indices with or without duplicates.

In the following example, the first select shows that there are seven employee rows matching the specified condition, however the subsequent cursor returns only two rows.

```
SQL> at 'f mf_personnel';
SQL> drop index EMP_EMPLOYEE_ID;
SQL> commit;
SQL> create unique index EMP_EMPLOYEE_ID
cont>     on EMPLOYEES (
cont>     EMPLOYEE_ID
cont>         asc)
cont>     type is SORTED ranked
cont>     node size 430
cont>     disable compression;
SQL> commit work;
SQL> declare transaction read write isolation level read committed;
SQL> select count(*) from employees
cont> where    employee_id > '00300'
cont> and      employee_id < '00400';

           7
1 row selected
SQL> declare t2 table cursor with hold preserve all for
cont> select    *
cont> from      employees
cont> where    employee_id > '00300'
```

```
cont> and      employee_id < '00400'
cont> order by employee_id desc;
SQL> open t2;
SQL> fetch t2;
 EMPLOYEE_ID   LAST_NAME        FIRST_NAME   MIDDLE_INITIAL
   ADDRESS_DATA_1              ADDRESS_DATA_2         CITY
      STATE   POSTAL_CODE   SEX    BIRTHDAY      STATUS_CODE
 00374         Andriola         Leslie       Q
   111 Boston Post Rd.                                Salisbury
      NH      03268         M      19-Mar-1955   1

SQL> commit;
SQL> fetch t2;
 EMPLOYEE_ID   LAST_NAME        FIRST_NAME   MIDDLE_INITIAL
   ADDRESS_DATA_1              ADDRESS_DATA_2         CITY
      STATE   POSTAL_CODE   SEX    BIRTHDAY      STATUS_CODE
 00369         Lapointe         Hope         NULL
   63 Union Square                                    Boscawen
      NH      03301         F      12-Mar-1948   1

SQL> commit;
SQL> fetch t2;
%RDB-E-STREAM_EOF, attempt to fetch past end of record stream
```

The problem only occurred if the retrieval was reset. A reset will occur if the transaction state has changed since the last fetch or if the index node being retrieved has been updated since the last fetch. In both of these cases, Oracle Rdb must re−compute its position in the index.

The problem will normally only be seen when *WITH HOLD* cursors are used and rows are fetched in separate transactions.

The problem can be avoided by retrieving the rows from the cursor in a single transaction that is either *READ ONLY* or *READ WRITE ISOLATION LEVEL SERIALIZABLE*.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.10 Restriction Removed for READ ONLY Transaction on Standby Database

Bug 3017015

In prior versions of Oracle Rdb, attempts to execute a transaction on a Standby database would fail if the SET TRANSACTION statement included a RESERVING ... FOR SHARED READ clause. This occurred when the Standby database was restored using the /TRANSACTION_MODE=READ_ONLY qualifier.

The following example shows the reported error.

```
SQL> attach 'filename mf_standby';
SQL> set transaction read only
cont>     reserving employees for shared read;
%RDB-E-BAD_TPB_CONTENT, invalid transaction parameters in the transaction parameter block
-RDMS-E-INVTRANOPT, the transaction option "SHARED READ" is not allowed
```

The /TRANSACTION_MODE qualifier is used to prevent read−write transactions executing on the Standby database during replication. However, this makes it impossible for some read−only applications to run against the standby database. This restriction is being relaxed for this release and the SHARED READ reserving clause will be ignored for READ ONLY transactions on the Standby database.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.11 Bugcheck at PSIINDEX$FIND_ENTS_EXACT + 54

Bug 2448304

When a combination of metadata operations was being performed in the same transaction, it was possible that a bugcheck could be generated with an exception similar to the following example.

```
***** Exception at 0106BC24 : PSIINDEX$FIND_ENTS_EXACT + 00000054
%SYSTEM−F−ACCVIO, access violation, reason mask=00,
  virtual address=0000000050000038, PC=000000000106BC24, PS=0000000B
```

This problem was caused by an incorrect use of internal memory management optimizations. This incorrect use can be identified by the unusual virtual adress of *0000000050000038*.

The problem can be avoided by reducing the number of metadata operations performed in a single transaction.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.12 AIJ Log Server Process May Loop or Bugcheck

Bugs 2651475 and 1756433

Under unknown but extremely rare conditions on busy databases where the After Image Journal (AIJ) Log Server process is enabled, the ALS process has been observed to enter a loop condition writing AIJ information to the AIJ file(s).

In the worst case, this problem could cause all available journal files to be filled with repeating data. If no remedial action were to be taken, this condition could cause the database to be shutdown and the AIJ journals to be considered inaccessable.

The database is not corrupted by this problem.

Stopping and restarting the ALS process will clear the looping condition even if the ALS process must be stopped using the STOP/ID command.

Stopping the ALS process will not impact production as AIJ writes will automatically revert to the non−ALS behaviour.

In Release 7.1.2, the behaviour of Oracle Rdb has been changed so that should this problem be detected, the ALS process will automatically shutdown producing a bugcheck dump file. This will prevent any danger of filling all available journals and will ensure that the database remains available.

ALS may be safely restarted immediately as the conditions that cause such a loop are resolved during recovery of the ALS process.

## 6.1.13 SYSTEM−F−NOIOCHAN Error

Bug 2540754

Repeated attaches/disconnects from a database eventually resulted in a SYSTEM−F−NOIOCHAN failure. This problem could occur only if either of the following two conditions was true:

- The database filename contained a concealed logical which included a search list and one of the items in the search list included a rooted directory name.
- A storage area filename contained a logical with a concealed, routed filename.

The following example shows a database definition containing both of the above conditions.

```
$ DEFINE/SYSTEM/EXEC/TRANS=(CONCEALED) UIS_ D1:[T.], D2:[T.]
$ DEFINE/SYSTEM/EXEC/TRANS=(CONCEALED) UIS_DEV_A D1:[T.]
$ DEFINE/SYSTEM/EXEC/TRANS=(CONCEALED) UIS_DEV_E D1:[T.]
$ DEFINE/SYSTEM/EXEC/TRANS=(CONCEALED) UIS_DEV_G D2:[T.]
$ SQL$
SQL> CREATE DATABASE FILENAME 'UIS_:[DB]FOO.RDB'
cont>  SEGMENTED STRING STORAGE AREA IS RDB$SYSTEM
cont>  DEFAULT STORAGE AREA IS FOOSYS
cont> CREATE STORAGE AREA RDB$SYSTEM
cont>  FILENAME 'UIS_DEV_A:[DB]FOO.RDA'
cont>  SNAPSHOT FILENAME 'UIS_DEV_A:[DB]FOO.SNP'
cont> CREATE STORAGE AREA UIS_SYSTEM_DATA
cont>  FILENAME 'UIS_DEV_G:[DB]FOOSYS.RDA'
cont>  SNAPSHOT FILENAME 'UIS_DEV_E:[DB]FOOSYS.SNP';
SQL> exit;
```

In the above example, a process that repeatedly attached to and disconnected from the FOO database would see its process open channel count increase by three for each set of attach/disconnects. This "channel leak" is associated with the database filename and each of the two storage areas. If the process repeated the attach/disconnect sequence enough times, eventually the open I/O channel count would be exhausted and the process would fail with SYSTEM−F−NOIOCHAN.

As a possible workaround, define the logicals without using concealed rooted filenames.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.14 ROLLBACK Not Appended to AIJ for Failed 2PC Transactions

Bug 2510623

If a process was a participant in a two−phase commit (2PC) transaction, and if any of the participants voted to commit the transaction, but before all of the participants voted one of the participants failed, the database recovery process (DBR) would neglect to write a rollback entry to the after−image journal (AIJ).

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.15 Bugcheck at LCK$MEMBIT_BLAST + 023C With Exception of COSI−F−IVLOCKID

Bug 2628150

Previously, it was possible for Rdb to bugcheck with an exception of COSI−F−IVLOCKID at LCK$MEMBIT_BLAST with a footprint similar to the following:

```
Exception at LCK$MEMBIT_BLAST + 0000023C
COSI−F−IVLOCKID, invalid lock id
Called from LCK$BLKAST + 00000084
Called from symbol not found
Called from LCK$MEMBIT_BLAST + 000001E4
Called from LCK$UNBIND + 000001D4
Called from KOD$UNBIND + 000003D4
Called from RDMS$$DETACH_DATABASE + 000006D8
```

This problem was due to a race condition when unbinding from the database while the database was being opened or closed on another node of the cluster. As a workaround, Oracle suggests explicitly opening databases with the RMU /OPEN command and leaving them open while available to users.

This problem has been corrected in Oracle Rdb Release 7.1.2. The race condition while unbinding from the database has been removed.

## 6.1.16 Bugcheck When Multiple Tables are Transitively Joined With IS NULL Filter

The following query with a transitive join between multiple tables bugchecks.

```
att 'file personnel';
select T0.SALARY_AMOUNT,
    T1.DEPARTMENT_CODE,
        T1.SUPERVISOR_ID,
        T2.EMPLOYEE_ID,
        T2.DEGREE,
        T2.COLLEGE_CODE,
        T2.DEGREE_FIELD
        from SALARY_HISTORY T0, JOB_HISTORY T1, DEGREES T2
          where T0.EMPLOYEE_ID = T1.EMPLOYEE_ID
                 and T1.EMPLOYEE_ID = T2.EMPLOYEE_ID
                 and T2.EMPLOYEE_ID is null;
```

This bugcheck occurs when multiple tables are transitively joined and the IS NULL filter is applied to the join column.

There is no known workaround.

This problem has been corrected in Oracle Rdb Release 7.1.2.

### 6.1.17 Unexpected Error From GRANT and REVOKE

Bug 2680837

In prior versions of Oracle Rdb, GRANT and REVOKE might fail if the table, sequence, module, function or procedure name was delimited and contained lowercase characters.

The following example shows the problem with wildcard GRANT on all tables. The table name was incorrectly converted to uppercase.

```
SQL> set quoting rules 'SQL92';
SQL> create table "little" ( i integer);
SQL> create table "little_BIG" (i integer);
SQL> create table BIG (i integer);
SQL>
SQL> grant all on table * to JONES;
%RDB−E−NO_META_UPDATE, metadata update failed
−RDB−E−OBSOLETE_METADA, request references metadata objects that no
longer exist
−RDMS−F−TABNOTDEF, relation LITTLE is not defined in database
```

A workaround for tables was to specify each table name explicitly.

```
SQL> set quoting rules 'SQL92';
SQL> grant all on table "little"     to JONES;
SQL> grant all on table "little_BIG" to JONES;
SQL> grant all on table BIG          to JONES;
```

This problem has been corrected in Oracle Rdb Release 7.1.2.

### 6.1.18 Process Can Hang After SYS$FORCEX Issued

Bug 2665186

Oracle Rdb processes could hang if the OpenVMS system service SYS$FORCEX was used to terminate a database application. For example, if an RMU /CLOSE was issued, some active database processes may have become hung in the LEF state. Note that the default for RMU /CLOSE is /ABORT=FORCEX. Another example of a command that uses the SYS$FORCEX facility is the "Terminate image" command that is available in the RMU /SHOW STATISTICS utility.

This problem has been corrected in Oracle Rdb Release 7.1.2. Oracle Rdb processes will no longer become hung due to the use of the SYS$FORCEX system service.

In the past, Oracle has recommended that SYS$FORCEX be used to terminate database processes since that could prevent a database recovery process (DBR) from being invoked to recover the failed database user. While this would often prevent a DBR from being invoked, it did not always prevent a DBR process from being started. A DBR process may need to be invoked if the forced exit request is received when there is an Oracle Rdb request active. This is due to limitations in the implementation of exit handlers in the OpenVMS operating system.

Forced exit requests are intercepted by using the OpenVMS exit handler facility. If there is no Oracle Rdb database request active at the time that the forced exit is received by the Oracle Rdb process, then the Oracle Rdb exit handler can gracefully rundown all Oracle Rdb activity. No DBR intervention is required. But, if the exit request comes in while an Oracle Rdb database request is active, then it is not possible to gracefully rundown Oracle Rdb.

An exit handler works effectively like an AST routine. That is, there is a good chance that when an exit request has been received some Oracle Rdb task has been interrupted. Oracle Rdb must handle the exit request when the exit handler is invoked, regardless of what activity is occurring at the time of the interrupt. Rundown activity cannot be postponed to a later time after the currently executing database request has finished. It is not possible for Oracle Rdb to unwind activity under way at the time of the forced exit request, so if Oracle Rdb finds that a database request is active at the time of the forced exit interrupt then no cleanup is attempted. The process fails abnormally and has to be recovered by a DBR.

## 6.1.19 Show Storage Map "Lists_Map" Partition Display Error

After a storage map for "lists" is created, a SHOW STORAGE MAP "LISTS_MAP" might incorrectly indicate that the lists are "Implicitly mapped to the default storage area" instead of displaying the desired information as a list of the storage areas actually used.

The following example illustrates the problem. The following is one such correct list.

```
SQL> show storage map lists_map

LISTS_MAP  For Lists Store clause:
   STORE lists in (RESUME_LISTS0, RESUME_LISTS1) for (newres.resume) FILL
   SEQUENTIALLY in (RESUME_LISTS2, RESUME_LISTS3, RESUME_LISTS5, RESUME_LISTS7)
   for (newres) FILL RANDOMLY in RESUME_LISTS -- this map uses the default list
   area as part of two mappings

 Partition information for lists map:
 Vertical Partition: VRP_P000
  Partition: (1) SYS_P00067
    Fill Sequentially
   Storage Area: RESUME_LISTS0
  Partition: (1) SYS_P00068
   Storage Area: RESUME_LISTS1
  Partition: (2) SYS_P00069
    Fill Randomly
```

The following is an example of the error:

```
SQL>   show storage map lists_map
     LISTS_MAP
 For Lists
 Store clause:    STORE lists
             in (RESUME_LISTS0,
                 RESUME_LISTS1) for (newres.resume) FILL SEQUENTIALLY
             in (RESUME_LISTS2,
                 RESUME_LISTS3,
                 RESUME_LISTS5,
                 RESUME_LISTS7) for (newres) FILL RANDOMLY
             in RESUME_LISTS
         -- this map uses the default list area as part of two mappings
```

```
 Partition information for lists map:
  Implicitly mapped to the default storage area
```

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.20 Incorrect Cardinalities Calculated by CREATE DATABASE

Bug 2421582

In prior versions of Oracle Rdb, CREATE DATABASE would calculate incorrect cardinalities for system relations RDB$FIELDS and RDB$RELATION_FIELDS.

The following example shows this behavior.

```
Old behaviour:
RELATION NAME                   STORED              ACTUAL
------------------------------------------------------------
RDB$FIELDS                      0                   41
RDB$RELATION_FIELDS             70                  432

Corrected  behaviour:
RELATION NAME                   STORED              ACTUAL
------------------------------------------------------------
RDB$FIELDS                      41                  41
RDB$RELATION_FIELDS             432                 432
```

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.21 ALTER INDEX Fails With the RDMS−F−NOT_LARDY Error

Bug 2722397

In prior releases of Oracle Rdb Release 7.1, attempts to use ALTER INDEX to TRUNCATE ALL PARTITIONS or REBUILD ALL PARTITIONS would fail with the error NOT_LARDY for indices that were not explicitly mapped.

The following example shows the error when an index is created on the JOBS table in the PERSONNEL database.

```
SQL> set flags 'index_stats';
SQL> alter index JOB_CODE_INDEX rebuild all partitions;
~Ai alter index "JOB_CODE_INDEX" (hashed=0, ordered=0)
~As locking table "JOBS"
~Ai truncate all partitions
~Ai truncated 0 partitions, skipped 0
~Ai shared larea − must destroy tree
%RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−F−NOT_LARDY, area for 4:733:2 not in proper ready mode
```

The error occurs because ALTER INDEX incorrectly thinks this index shares the logical area with another index. The DROP INDEX and CREATE INDEX commands must be used to rebuild this type of index.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.22 Cannot Drop Close Timer Database

Bug 2711207

If a database has a wait for close timer defined, it cannot be dropped. For example:

```
SQL> create database filename foo open automatic (wait 1 minutes for close);
SQL> disconnect all;
SQL> $wait 0:1:30
SQL> drop database filename foo;
%RDB−F−SYS_REQUEST, error from system services request
−RDMS−F−FILACCERR, error opening database root file DEV:[DIR]FOO.RDB;1
−SYSTEM−W−ACCONFLICT, file access conflict
SQL>
```

To avoid this problem, disable the close timer prior to dropping the database.

```
SQL> alter database filename foo open automatic (wait 0 minutes for close);
SQL> drop database filename foo;
SQL>
```

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.23 Outer ORDER BY Ignored in Favor of Inner ORDER BY Clause

Bug 2649678

The following query returned the results in the wrong sort order. The query has two explicit sorts, an outer sort on the birthday column in ascending order and a sort in the nested select clause on the birthday column in descending sort order. The query results were being returned in descending order of the birthday column.

```
select o.* from
  (select distinct i.last_name, i.first_name, i.birthday
  from employees i order by birthday desc limit to 10 rows) o
order by birthday asc;
```

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.24 Query Using IN Clause With 16 Filter Predicates Runs Slower

Bug 2634849

The following query using an IN clause with 16 filter predicates takes much longer than a query that uses 15 filter predicates where dynamic OR strategy is applied.

```
 SELECT COUNT(*)
 FROM
 (( SELECT
    OPERATION_NO, PLATE_ID, FINISH_DATIME
    ,BATCH_ID
      FROM LCD_PLATE_HISTORY
      WHERE   OPERATION_NO = '3040' AND PLATE_ID IN
  ('BNL70826-03','BNL70826-04','BNL70826-05','BNL70826-06','BNL70826-10',
   'BNL70826-11','BNL70826-12','BNL70826-13','BNL70826-14','BNL70826-15',
   'BNL70826-18','BNL70828-01','BNL70828-02','BNL70828-03','BNL70828-04',
   'BNL70828-05'
   ))
  AS TFT_H
  ( OPERATION_NO, PLATE_ID, FINISH_DATIME
    ,BATCH_ID
   )
 INNER JOIN
 ( SELECT OPERATION_NO
    FROM LCD_PROCESS_DATA_ITEMS
    WHERE   PROCESS_DATA_TYPE ='A' AND OPERATION_NO = '3040' )
      AS TFT_D1 ( OPERATION_NO)
         ON TFT_H.OPERATION_NO = TFT_D1.OPERATION_NO
 )
 LEFT OUTER JOIN
 ( SELECT   PLATE_ID, OPERATION_NO
   FROM LCD_PLATE_PROCESS_A)
   AS TFT_A1 ( PLATE_ID, OPERATION_NO )
  ON TFT_H.OPERATION_NO = TFT_A1.OPERATION_NO
        AND TFT_H.PLATE_ID = TFT_A1.PLATE_ID
;
Tables:
  0 = LCD_PLATE_HISTORY
  1 = LCD_PROCESS_DATA_ITEMS
  2 = LCD_PLATE_PROCESS_A
Aggregate: 0:COUNT (*)
Cross block of 2 entries        (Left Outer Join)
  Cross block entry 1
    Cross block of 2 entries
      Cross block entry 1
        Merge of 1 entries
          Merge block entry 1
          Conjunct: (1.PROCESS_DATA_TYPE = 'A') AND (1.OPERATION_NO = '3040')
          Leaf#01 BgrOnly 1:LCD_PROCESS_DATA_ITEMS Card=1
            Bool: 1.OPERATION_NO = '3040'
            BgrNdx1 LCD_PROCESS_DATA_ITEMS_IDX_S2 [1:1] Fan=13
              Keys: 1.OPERATION_NO = '3040'
      Cross block entry 2
        Merge of 1 entries
          Merge block entry 1
          Conjunct: 0.OPERATION_NO = 1.OPERATION_NO
          Leaf#02 BgrOnly 0:LCD_PLATE_HISTORY Card=505000
            Bool: (0.OPERATION_NO = '3040') AND ((0.PLATE_ID = 'BNL70826-03')
                   OR (0.PLATE_ID = 'BNL70826-04') OR (0.PLATE_ID =
                  'BNL70826-05') OR (0.PLATE_ID = 'BNL70826-06') OR (0.PLATE_ID
                   = 'BNL70826-10') OR (0.PLATE_ID = 'BNL70826-11') OR (
```

```
                  0.PLATE_ID = 'BNL70826-12') OR (0.PLATE_ID = 'BNL70826-13')
                   OR (0.PLATE_ID = 'BNL70826-14') OR (0.PLATE_ID =
                  'BNL70826-15') OR (0.PLATE_ID = 'BNL70826-18') OR (0.PLATE_ID
                   = 'BNL70828-01') OR (0.PLATE_ID = 'BNL70828-02') OR (
                  0.PLATE_ID = 'BNL70828-03') OR (0.PLATE_ID = 'BNL70828-04')
                   OR (0.PLATE_ID = 'BNL70828-05'))
            BgrNdx1 LCD_PLATE_HISTORY_IDX_S1 [1:1] Fan=9    <== See Note1
              Keys: 0.OPERATION_NO = 1.OPERATION_NO
              Bool: (0.OPERATION_NO = '3040') AND ((0.PLATE_ID = 'BNL70826-03')
                     OR (0.PLATE_ID = 'BNL70826-04') OR (0.PLATE_ID =
                    'BNL70826-05') OR (0.PLATE_ID = 'BNL70826-06') OR (
                    0.PLATE_ID = 'BNL70826-10') OR (0.PLATE_ID = 'BNL70826-11')
                     OR (0.PLATE_ID = 'BNL70826-12') OR (0.PLATE_ID =
                    'BNL70826-13') OR (0.PLATE_ID = 'BNL70826-14') OR (
                    0.PLATE_ID = 'BNL70826-15') OR (0.PLATE_ID = 'BNL70826-18')
                     OR (0.PLATE_ID = 'BNL70828-01') OR (0.PLATE_ID =
                    'BNL70828-02') OR (0.PLATE_ID = 'BNL70828-03') OR (
                    0.PLATE_ID = 'BNL70828-04') OR (0.PLATE_ID = 'BNL70828-05'))
  Cross block entry 2
    Merge of 1 entries
      Merge block entry 1
      Conjunct: (0.OPERATION_NO = 2.OPERATION_NO) AND (0.PLATE_ID = 2.PLATE_ID)
      Index only retrieval of relation 2:LCD_PLATE_PROCESS_A
        Index name  LCD_PLATE_PROCESS_A_IDX_S1 [2:2]
          Keys: (0.PLATE_ID = 2.PLATE_ID) AND (0.OPERATION_NO = 2.OPERATION_NO)

           5
1 row selected
```

Note1: Incorrect retrieval via LCD_PLATE_HISTORY_IDX_S1 [1:1] is applied instead of dynamic OR strategy.

The query works applying dynamic OR strategy if one of the columns in the select list is commented out.

```
 SELECT COUNT(*)
 FROM
 (( SELECT
    OPERATION_NO, PLATE_ID, FINISH_DATIME
!   ,BATCH_ID
      FROM LCD_PLATE_HISTORY
      WHERE   OPERATION_NO = '3040' AND PLATE_ID IN
  ('BNL70826-03','BNL70826-04','BNL70826-05','BNL70826-06','BNL70826-10',
   'BNL70826-11','BNL70826-12','BNL70826-13','BNL70826-14','BNL70826-15',
   'BNL70826-18','BNL70828-01','BNL70828-02','BNL70828-03','BNL70828-04',
   'BNL70828-05'
   ))
  AS TFT_H
  ( OPERATION_NO, PLATE_ID, FINISH_DATIME
!   ,BATCH_ID
   )
 INNER JOIN
 ( SELECT OPERATION_NO
    FROM LCD_PROCESS_DATA_ITEMS
    WHERE   PROCESS_DATA_TYPE ='A' AND OPERATION_NO = '3040' )
     AS TFT_D1 ( OPERATION_NO)
      ON TFT_H.OPERATION_NO = TFT_D1.OPERATION_NO
 )
 LEFT OUTER JOIN
 ( SELECT   PLATE_ID, OPERATION_NO
```

```
   FROM LCD_PLATE_PROCESS_A)
   AS TFT_A1 ( PLATE_ID, OPERATION_NO )
  ON TFT_H.OPERATION_NO = TFT_A1.OPERATION_NO
         AND TFT_H.PLATE_ID = TFT_A1.PLATE_ID
;
Tables:
  0 = LCD_PLATE_HISTORY
  1 = LCD_PROCESS_DATA_ITEMS
  2 = LCD_PLATE_PROCESS_A
Aggregate: 0:COUNT (*)
Cross block of 2 entries        (Left Outer Join)
  Cross block entry 1
    Cross block of 2 entries
      Cross block entry 1
        Merge of 1 entries
          Merge block entry 1
          Conjunct: (1.PROCESS_DATA_TYPE = 'A') AND (1.OPERATION_NO = '3040')
          Leaf#01 BgrOnly 1:LCD_PROCESS_DATA_ITEMS Card=1
            Bool: 1.OPERATION_NO = '3040'
            BgrNdx1 LCD_PROCESS_DATA_ITEMS_IDX_S2 [1:1] Fan=13
              Keys: 1.OPERATION_NO = '3040'
      Cross block entry 2
        Merge of 1 entries
          Merge block entry 1
          Conjunct: 0.OPERATION_NO = 1.OPERATION_NO
          Leaf#02 NdxOnly 0:LCD_PLATE_HISTORY Card=505000
            Bool: (0.OPERATION_NO = '3040') AND ((0.PLATE_ID = 'BNL70826-03')
                   OR (0.PLATE_ID = 'BNL70826-04') OR (0.PLATE_ID =
                  'BNL70826-05') OR (0.PLATE_ID = 'BNL70826-06') OR (0.PLATE_ID
                   = 'BNL70826-10') OR (0.PLATE_ID = 'BNL70826-11') OR (
                  0.PLATE_ID = 'BNL70826-12') OR (0.PLATE_ID = 'BNL70826-13')
                   OR (0.PLATE_ID = 'BNL70826-14') OR (0.PLATE_ID =
                  'BNL70826-15') OR (0.PLATE_ID = 'BNL70826-18') OR (0.PLATE_ID
                   = 'BNL70828-01') OR (0.PLATE_ID = 'BNL70828-02') OR (
                  0.PLATE_ID = 'BNL70828-03') OR (0.PLATE_ID = 'BNL70828-04')
                   OR (0.PLATE_ID = 'BNL70828-05'))
            FgrNdx  LCD_PLATE_HISTORY_IDX_S1 [(2:2)16] Fan=9
              Keys: r0: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                        'BNL70828-05')
                    r1: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                        'BNL70828-04')
                    r2: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                        'BNL70828-03')
                    r3: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                        'BNL70828-02')
                    r4: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                        'BNL70828-01')
                    r5: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                        'BNL70826-18')
                    r6: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                        'BNL70826-15')
                    r7: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                        'BNL70826-14')
                    r8: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                        'BNL70826-13')
                    r9: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                        'BNL70826-12')
                    r10: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                        'BNL70826-11')
                    r11: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                        'BNL70826-10')
```

```
                r12: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                    'BNL70826-06')
                r13: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                    'BNL70826-05')
                r14: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                    'BNL70826-04')
                r15: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                    'BNL70826-03')
        BgrNdx1 LCD_PLATE_HISTORY_IDX_S4 [(2:2)16] Fan=9
          Keys: r0: (0.PLATE_ID = 'BNL70828-05') AND (0.OPERATION_NO =
                    '3040')
                r1: (0.PLATE_ID = 'BNL70828-04') AND (0.OPERATION_NO =
                    '3040')
                r2: (0.PLATE_ID = 'BNL70828-03') AND (0.OPERATION_NO =
                    '3040')
                r3: (0.PLATE_ID = 'BNL70828-02') AND (0.OPERATION_NO =
                    '3040')
                r4: (0.PLATE_ID = 'BNL70828-01') AND (0.OPERATION_NO =
                    '3040')
                r5: (0.PLATE_ID = 'BNL70826-18') AND (0.OPERATION_NO =
                    '3040')
                r6: (0.PLATE_ID = 'BNL70826-15') AND (0.OPERATION_NO =
                    '3040')
                r7: (0.PLATE_ID = 'BNL70826-14') AND (0.OPERATION_NO =
                    '3040')
                r8: (0.PLATE_ID = 'BNL70826-13') AND (0.OPERATION_NO =
                    '3040')
                r9: (0.PLATE_ID = 'BNL70826-12') AND (0.OPERATION_NO =
                    '3040')
                r10: (0.PLATE_ID = 'BNL70826-11') AND (0.OPERATION_NO =
                    '3040')
                r11: (0.PLATE_ID = 'BNL70826-10') AND (0.OPERATION_NO =
                    '3040')
                r12: (0.PLATE_ID = 'BNL70826-06') AND (0.OPERATION_NO =
                    '3040')
                r13: (0.PLATE_ID = 'BNL70826-05') AND (0.OPERATION_NO =
                    '3040')
                r14: (0.PLATE_ID = 'BNL70826-04') AND (0.OPERATION_NO =
                    '3040')
                r15: (0.PLATE_ID = 'BNL70826-03') AND (0.OPERATION_NO =
                    '3040')
  Cross block entry 2
    Merge of 1 entries
      Merge block entry 1
      Conjunct: (0.OPERATION_NO = 2.OPERATION_NO) AND (0.PLATE_ID = 2.PLATE_ID)
      Index only retrieval of relation 2:LCD_PLATE_PROCESS_A
        Index name  LCD_PLATE_PROCESS_A_IDX_S1 [2:2]
          Keys: (0.PLATE_ID = 2.PLATE_ID) AND (0.OPERATION_NO = 2.OPERATION_NO)

           5
1 row selected
```

As a potential workaround, the query works if the dynamic strategy is disabled by setting the SQL flag 'MAX_STABILITY' or defining the logical RDMS$MAX_STABILITY as Y.

```
set flags 'MAX_STABILITIY';
!execute the above same query
@bug.sql
Tables:
  0 = LCD_PLATE_HISTORY
```

```
    1 = LCD_PROCESS_DATA_ITEMS
    2 = LCD_PLATE_PROCESS_A
  Aggregate: 0:COUNT (*)
  Cross block of 2 entries        (Left Outer Join)
    Cross block entry 1
      Cross block of 2 entries
        Cross block entry 1
          Merge of 1 entries
            Merge block entry 1
            Conjunct: 1.PROCESS_DATA_TYPE = 'A'
            Conjunct: 1.OPERATION_NO = '3040'
            Get     Retrieval by index of relation 1:LCD_PROCESS_DATA_ITEMS
              Index name  LCD_PROCESS_DATA_ITEMS_IDX_S2 [1:1]
                Keys: 1.OPERATION_NO = '3040'
        Cross block entry 2
          Merge of 1 entries
            Merge block entry 1
            Conjunct: 0.OPERATION_NO = 1.OPERATION_NO
            Conjunct: (0.OPERATION_NO = '3040') AND ((0.PLATE_ID = 'BNL70826-03')
                       OR (0.PLATE_ID = 'BNL70826-04') OR (0.PLATE_ID =
                      'BNL70826-05') OR (0.PLATE_ID = 'BNL70826-06') OR (
                      0.PLATE_ID = 'BNL70826-10') OR (0.PLATE_ID = 'BNL70826-11')
                       OR (0.PLATE_ID = 'BNL70826-12') OR (0.PLATE_ID =
                      'BNL70826-13') OR (0.PLATE_ID = 'BNL70826-14') OR (
                      0.PLATE_ID = 'BNL70826-15') OR (0.PLATE_ID = 'BNL70826-18')
                       OR (0.PLATE_ID = 'BNL70828-01') OR (0.PLATE_ID =
                      'BNL70828-02') OR (0.PLATE_ID = 'BNL70828-03') OR (
                      0.PLATE_ID = 'BNL70828-04') OR (0.PLATE_ID = 'BNL70828-05'))
            Get     Retrieval by index of relation 0:LCD_PLATE_HISTORY
              Index name  LCD_PLATE_HISTORY_IDX_S1 [(2:2)16]
                Keys: r0: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                          'BNL70828-05')
                      r1: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                          'BNL70828-04')
                      r2: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                          'BNL70828-03')
                      r3: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                          'BNL70828-02')
                      r4: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                          'BNL70828-01')
                      r5: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                          'BNL70826-18')
                      r6: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                          'BNL70826-15')
                      r7: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                          'BNL70826-14')
                      r8: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                          'BNL70826-13')
                      r9: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                          'BNL70826-12')
                      r10: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                          'BNL70826-11')
                      r11: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                          'BNL70826-10')
                      r12: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                          'BNL70826-06')
                      r13: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                          'BNL70826-05')
                      r14: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
                          'BNL70826-04')
                      r15: (0.OPERATION_NO = '3040') AND (0.PLATE_ID =
```

```
                        'BNL70826-03')
              Bool: 0.OPERATION_NO = '3040'
  Cross block entry 2
    Merge of 1 entries
      Merge block entry 1
      Conjunct: (0.OPERATION_NO = 2.OPERATION_NO) AND (0.PLATE_ID = 2.PLATE_ID)
      Index only retrieval of relation 2:LCD_PLATE_PROCESS_A
        Index name  LCD_PLATE_PROCESS_A_IDX_S1 [2:2]
          Keys: (0.PLATE_ID = 2.PLATE_ID) AND (0.OPERATION_NO = 2.OPERATION_NO)

           5
1 row selected
```

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.25 PIOUTL$BUILD_IORB Bugcheck Adding Mixed Format Storage Area

Bug 2753947

When adding storage areas to a database when the new storage area has a page size smaller than the page size of any existing storage area, it was possible for memory corruption to occur. The symptoms of this corruption could include bugcheck dumps with an ACCVIO exception at PIOUTL$BUILD_IORB.

The following command sequence illustrates one possible failure case.

```
$ SQL
    CREATE DATABASE FILENAME 'TESTDB'
    NUMBER OF CLUSTER NODES 1              BUFFER SIZE IS 24 BLOCKS
    NUMBER OF BUFFERS 50                   SHARED MEMORY IS SYSTEM
    RESERVE 200 STORAGE AREAS              RESERVE 110 JOURNALS
    DEFAULT STORAGE AREA IS RDB$SYSTEM

    CREATE STORAGE AREA RDB$SYSTEM FILENAME 'TESTDB_SYSTEM_AREA.RDA'
        PAGE SIZE IS 6 BLOCKS              ALLOCATION IS 1000 PAGES
        EXTENT IS (MINIMUM 5000, MAXIMUM 10000, PERCENT GROWTH 20)
        SNAPSHOT FILENAME 'TESTDB_SYSTEM_AREA.SNP'
        SNAPSHOT ALLOCATION IS 100 PAGES
        SNAPSHOT EXTENT IS (MINIMUM 10, MAXIMUM 10, PERCENT GROWTH 1);

    EXIT;
$ SQL
    ALTER DATABASE FILENAME TESTDB
    ADD STORAGE AREA A_MIXED_AREA FILENAME 'A_MIXED_AREA.RDA'
        PAGE FORMAT IS MIXED
        PAGE SIZE IS 3 BLOCKS              ALLOCATION IS 20 PAGES
        EXTENT IS (MINIMUM 1000, MAXIMUM 1000, PERCENT GROWTH 20)
        SNAPSHOT FILENAME 'A_MIXED_AREA.SNP'
        SNAPSHOT ALLOCATION IS 10 PAGES
        SNAPSHOT EXTENT IS (MINIMUM 10, MAXIMUM 10, PERCENT GROWTH 1);
    EXIT;
```

This problem has been corrected in Oracle Rdb Release 7.1.2. The potential failure was due to incorrect "maximum pages in buffer" calculations being performed after the new area was added with the smallest page size in the database. The new storage area is now added in a strictly "offline" mode to prevent inaccurate page size information from being used.

## 6.1.26 Unexpected Bugcheck During CREATE SEQUENCE

Bug 2753954

In prior versions of Oracle Rdb, attempts to use CREATE SEQUENCE in a READ ONLY transaction would bugcheck if the database had snapshots disabled.

```
Alpha OpenVMS 7.3
Oracle Rdb Server V7.1−04
Got a RDSBUGCHK.DMP
COSI−F−BUGCHECK, internal consistency failure
Exception occurred at SEQ$CREATE_SEQUENCE + 00000280
Called from RDMS$$CREATE_SEQUENCE_INFO + 0000033C
Called from RDMS$$RELEASE_DDL_VM_HNDLR + 00000DD8
```

The following example shows an example script being executed.

```
SQL> set flags 'transaction';
SQL>
SQL> set transaction read only;
~T Compile transaction (1) on db: 1
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_READ (read only)
~T Start_transaction (1) on db: 1, db count=1
~T Snapshots are disabled, READ ONLY converted to READ WRITE
SQL>
SQL> create sequence DEPT_ID_SEQ;
%RDMS−I−BUGCHKDMP, generating bugcheck dump file USER2:[TESTING]RDSBUGCHK.DMP;
SQL> rollback;
```

This problem has been corrected in Oracle Rdb Release 7.1.2. Rdb now checks for the READ ONLY transaction prior to creating the sequence in the Rdb database root file. An exception is now raised as shown in this example.

```
SQL> create sequence DEPT_ID_SEQ;
%RDB−E−NO_META_UPDATE, metadata update failed
−RDB−E−READ_ONLY_TRANS, attempt to update during a read−only transaction
```

## 6.1.27 Bugcheck on Dropping of an Empty Sorted Ranked Index

Bug 2772919

A problem in the checking of index node integrity on the dropping of a sorted ranked index caused a bugcheck to be incorrectly raised.

The bugcheck shows the following exception:

```
***** Exception at 00DAA9C8 : PSII2DESTROYDUPCHAINS + 00000108
%COSI−F−BUGCHECK, internal consistency failure
```

This problem only occurs when a sorted ranked index is created on an empty table and then dropped prior to any records being inserted into the table.

A possible workaround is to insert a dummy record into the indexed table prior to dropping the index.

This problem has been corrected in Oracle Rdb Release 7.1.2.

### 6.1.28 Index Estimation May Not Estimate the Most Useful Index First

Bug 650980

When a dynamic tactic is chosen for query execution, Oracle Rdb may perform estimation each time the query executes to determine the most useful index for retrieval. Estimation involves examining the index to determine which of multiple indexes would be the most useful for retrieving the required data. If an index examined during estimation would return zero rows, the execution of that retrieval is terminated since it is known that the query will return zero rows. This is termed zero shortcut.

In the past, when a zero shortcut is performed, Rdb did not re−order the background indexes. This meant that the indexes would be estimated in the same sequence on the next execution. Should a query execute many times over, where the second background index caused zero shortcut, the effort to estimate the first index every time is wasted and could cause excessive IO.

In the following example, a portion of the execution trace for a query shows how background index 1 estimates at 1 row but background index 2 gives a precise estimate of zero causing zero shortcut. In this case, the IO needed to estimate background index 1 is wasted.

```
~E#0005.04(1) Estim  Ndx:Lev/Seps/DBKeys 1:1/1/1 2:1/0/0 ZeroShortcut
~E#0005.04(2) Estim  Ndx:Lev/Seps/DBKeys 1:1/1/1 2:1/0/0 ZeroShortcut
~E#0005.04(3) Estim  Ndx:Lev/Seps/DBKeys 1:1/1/1 2:1/0/0 ZeroShortcut
~E#0005.04(4) Estim  Ndx:Lev/Seps/DBKeys 1:1/1/1 2:1/0/0 ZeroShortcut
~E#0005.04(5) Estim  Ndx:Lev/Seps/DBKeys 1:1/1/1 2:1/0/0 ZeroShortcut
~E#0005.04(6) Estim  Ndx:Lev/Seps/DBKeys 1:1/1/1 2:1/0/0 ZeroShortcut
~E#0005.04(7) Estim  Ndx:Lev/Seps/DBKeys 1:1/1/1 2:1/0/0 ZeroShortcut
```

Rdb now forces the index with the zero estimate to be the first index estimated on the subsequent execution. This eliminates IO wasted on estimating other indices where one index repeatedly zero shortcuts.

The corrected execution trace will show the indexes in the new order.

```
~E#0006.04(1) Estim   Index/Estimate 2/0 1/1 ZeroShortcut
~E#0006.04(2) Estim   Index/Estimate 2/0 1_1 ZeroShortcut
~E#0006.04(3) Estim   Index/Estimate 2/0 1_1 ZeroShortcut
~E#0006.04(4) Estim   Index/Estimate 2/0 1_1 ZeroShortcut
~E#0006.04(5) Estim   Index/Estimate 2/0 1_1 ZeroShortcut
~E#0006.04(6) Estim   Index/Estimate 2/0 1_1 ZeroShortcut
~E#0006.04(7) Estim   Index/Estimate 2/0 1_1 ZeroShortcut
```

Notice how on the second and subsequent executions, the background index 1 shows an underscore "_" character. This indicates that estimation was not performed for that index.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.29 Premature Switch to Sequential Retrieval in the Dynamic Optimizer

Bug 1335370

When Oracle Rdb chooses to use the dynamic optimizer, it uses an estimate of the table cardinality to limit how much of an index is read before it is considered too costly. If this limit is reached, Rdb assumes it is less costly to read the data using sequential retrieval than to pursue the index lookup.

In the following example, a user attaches to a database and queries a table containing no rows. On first reference to the table, the user will record the tables current cardinality. A second user then inserts approximately 60,000 rows into the table. Because the first user has no way of identifying the presence of the new rows, when the table is queried again the dynamic optimizer prematurely switches to sequential retrieval. This causes a large amount of IO to be performed during the FIN phase.

```
~S#0005
Tables:
  0 = RVOPERCONS
Firstn: 15
Sort: <mapped field>(a)
Aggregate: 0:COUNT (*)
           1:SUM (0.OPC_NUMTITU)
           2:SUM (0.OPC_IMPORTE_EFECT)
           3:MIN (0.OPC_NUMORD_OPER)
Sort: 0.OPC_VALORRV(a), 0.OPC_PRECIORV(a)
Leaf#01 BgrOnly 0:RVOPERCONS Card=0
  Bool: (0.OPC_VALORRV = 'TEF') AND (0.OPC_NUMORD_OPER >= 0) AND (((
        0.OPC_CODSOCNS_RECC = '9820') AND (0.OPC_FECHA_ORDCOMP = '20000605')
         AND (0.OPC_NUMORD_COMP = 60055)) OR ((0.OPC_CODSOCNS_RECV = '9820')
         AND (0.OPC_FECHA_ORDVENT = '20000605') AND (0.OPC_NUMORD_VENT = 60055))
        )
  BgrNdx1 IXOPC1SU [2:1] Fan=14
    Keys: (0.OPC_VALORRV = 'TEF') AND (0.OPC_NUMORD_OPER >= 0)
~E#0005.01(1) BgrNdx1 FtchLim  DBKeys=1016  Fetches=4+42  RecsOut=0
~E#0005.01(1) Fin     Seq      DBKeys=60451  Fetches=0+2329  RecsOut=26
```

To avoid the problem, the user can do the following:

- ♦ Detach and re−attach to the database. When the user re−attaches, the updated table cardinality will be retrieved from the metadata that will include the newly inserted rows.
- ♦ Use a query outline. A query outline with *EXECUTION OPTIONS NONE* will prevent the use of the dynamic optimizer which should ensure a static index lookup that will not switch to sequential.

During the estimation phase of dynamic execution, Rdb attempts to update the cardinality used to determine the limit for the switch to sequential retrieval. It does this by using index depth and the number of entries in the root node of a *TYPE IS SORTED* index to estimate the number of rows in a table. However, this calculation did not take place if there was only one background index or the index was *TYPE IS SORTED RANKED* or the index was partitioned.

This problem has been corrected by several changes in the dynamic optimizer.

- For ranked indexes, Rdb now calculates the estimate of rows based on the ranking information in the root node of the index during index estimation.
- Estimation will be attempted even if there is only one background index.
- Estimates for ranked indexes are considerably more reliable and are used in preference to estimates from sorted (non−ranked) indexes.

In addition, the execution trace information has been enhanced to enable display of the dynamic cardinality updates using *SET FLAGS 'EXECUTION,DETAIL(1)'*.

The following example shows the new execution trace. Since the strategy is the same as the previous example, it has been omitted in this example.

```
~Estim  Ndx1 Ranked: Nodes=155, Min=0, Est=3110 IO=1
~Estim  RLEAF Cardinality=  4.3821000E+04
~E#0004.01(1) Estim   Index/Estimate 1/3110
~E#0004.01(1) BgrNdx1 EofBuf   DBKeys=1024  Fetches=3+42  RecsOut=0
~E#0004.01(1) BgrNdx1 EofBuf   DBKeys=2048* Fetches=0+41  RecsOut=0
~E#0004.01(1) BgrNdx1 EofBuf   DBKeys=3072* Fetches=0+40  RecsOut=0
~E#0004.01(1) BgrNdx1 EofData  DBKeys=4058* Fetches=0+38  RecsOut=0 #Bufs=1017
~E#0004.01(1) Fin     TTbl     DBKeys=4058  Fetches=0+1017  RecsOut=26
```

Notice how the new debug information shows the new cardinality estimate for this dynamic strategy is 4.3821000E+04 or 43,821 rows. This is reasonably close to the actual table cardinality of 60,000 rows. Because the table cardinality for this dynamic tactic is now more accurate, execution does not switch to sequential retrieval and the index scan executes to completion resulting in a significant performance improvement in terms of IO and elapsed time.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.30 Poor Performance From Bulk Loads After 7.1.0.1

Bug 2782338

In Oracle Rdb Release 7.1.0.1, the asynchronous batch write (ABW) facility was changed to no longer write large batches of pages at a time. While this change worked fine for applications that would randomly update some buffers in the buffer pool and not others, it did impact the performance of bulk load and other update intensive operations. Not doing large batches increased the number of recovery unit journal (RUJ) file writes done in a single transaction causing more I/O to be required for a transaction.

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.2.

Oracle Rdb will again write a group of modified buffers when the ABW facility is triggered. The number of buffers written will be as much as specified or defaulted for the database Asynchronous Batch Write options "MAXIMUM BUFFER COUNT IS buffer−count" parameter. This number is displayed in the following RMU /DUMP /HEADER output as "Maximum batch size is 4 buffers":

```
   − Asynchronous batch−write is enabled
       Clean buffer count is 5
       Maximum batch size is 4 buffers
```

Fewer pages may be written if the oldest buffers in the buffer pool are not all modified.

## 6.1.31 RDMRLE Image Not Always Supplied During Installation

Bug 2799280

Depending on other versions of Oracle Rdb being previously installed on the system, the RDMRLE%%.EXE image would not be supplied by the installation procedure.

As a potential workaround, the RDMRLE%%.EXE image can be manually extracted from the kit savesets and moved to SYS$COMMON:[SYSLIB].

This problem has been corrected in Oracle Rdb Release 7.1.2. The RDMRLE%%.EXE image is now correctly supplied by the installation procedure.

## 6.1.32 SORT Consumes All Available Memory

With ever larger databases and systems using extremely large Working Set parameters, SORT's input parameters cause it to allocate all of the available memory for use in an in−memory sort. While this might appear to be a good idea from a performance perspective, it seldom is better than using a much smaller quantity of memory along with some sort work files. Such allocation also severely limits the performance of other software components. We have limited SORT's allocation of memory for all future versions of Rdb. In addition, it may be to the user's advantage to further constrain sort by using the logical RDMS$BIND_SORT_MEMORY_MAX_BYTES as in the following example.

```
$ DEFINE RDMS$BIND_SORT_MEMORY_MAX_BYTES 1000000
```

We use 1 million as an example. Different numbers may provide better performance depending on the application.

In addition, we have added information to the bugcheck to provide better diagnosis of this problem.

A workaround for this problem is to use the logical name as described above. Some experimentation may be needed to select an optimum number for the parameter.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.33 Unexpected Failure of the IDENTITY Clause in CREATE TABLE

In prior releases of Oracle Rdb 7.1, the use of IDENTITY would sometimes fail. This occurred when the name of the column was the same as the domain used for the type.

The following example shows the error:

```
SQL> create table PRODUCTS
cont>     (product_id         PRODUCT_ID identity primary key,
cont>      product_name       PRODUCT_NAME,
```

```
cont>      unit_price         MONEY,
cont>      unit_name          char (10));
%RDB−E−NO_META_UPDATE, metadata update failed
−RDB−E−SEQNONEXT, The next value for the sequence "PRODUCTS" is not available
```

This problem has been corrected in Oracle Rdb Release 7.1.2. Rdb now correctly processes the IDENTITY attribute in this case.

## 6.1.34 Unexpected Failure of INSERT for Table With IDENTITY After an IMPORT

In prior versions of Oracle Rdb 7.1, IDENTITY columns were not correctly imported. Although the IMPORT completes successfully, attempts to use the table with the IDENTITY column fail. The error is shown in this example.

```
SQL> import data from SQL_COLUMN_IDENTITY_4.RBR file xx;
SQL> ! Add new order
SQL> insert into ORDERS
cont>     values ((select customer_id from CUSTOMERS
cont>              where customer_name = 'ian'));
%RDB−E−SEQNONEXT, The next value for the sequence "ORDERS" is not available
```

The problem is that the IMPORT DATABASE statement did not correctly mark the column as an IDENTITY column.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.35 Unexpected Privileges Required Using VLM or SSB Features with OpenVMS Galaxy Support Enabled

Previously, processes accessing or creating row caches using the VLM (Very Large Memory) or SSB (System Space Buffers) features in an OpenVMS Galaxy environment were required to have the PRMGBL and SYSGBL privileges enabled.

This problem has been corrected in Oracle Rdb Release 7.1.2. During VLM or SSB creation or mapping in a Galaxy Environment, Oracle Rdb explicitly enables the PRMGBL and SYSGBL privileges.

## 6.1.36 Bugchecks at PIO$FETCH + 00000360

Bug 2846828

Oracle Rdb would occasionally bugcheck with the following exception when attempts to upgrade a lock mode on a database storage area resulted in a lock conflict.

```
***** Exception at 0057CF60 : PIO$FETCH + 00000360
%COSI−F−BUGCHECK, internal consistency failure
```

There is no workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.37 DBR Bugchecks at DBR$DDTM_RESOLVE + 000003F4

The database recovery process (DBR) would sometimes bugcheck with the following failure:

```
***** Exception at 0005EC94 : DBR$DDTM_RESOLVE + 000003F4
%COSI−F−BUGCHECK, internal consistency failure
```

This bugcheck would occur when the OpenVMS system service $GETDTIW would return an undocumented transaction state value (11, or "IN_DOUBT") for an unresolved transaction. The DBR would fail since it was not expecting that transaction state.

Subsequent attempts to open or attach to the database would usually succeed. The second query using the $GETDTIW would typically return an expected state value.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.38 Spurious CHECKSUM Errors Reading READ ONLY Areas with Global Buffers

In Oracle Rdb Release 7.1.0.4 through Release 7.1.1, it was possible for processes to report spurious checksum errors if an area was set to READ ONLY and the global buffer feature was enabled. For example, OPCOM messages similar to the following might be reported:

```
%%%%%%%%%%%  OPCOM  30−MAR−2003 15:14:17.35  %%%%%%%%%%%
Message from user RDB_RANDOM on NODE1
Oracle Rdb V7.1−1 Event Notification for Database
DEV:[DIR]MF_PERSONNEL.RDB;1

Page 10:273 checksum error − computed C64F3171, page contained C64E3171;
retrying disk read
```

The problem can be avoided by changing the storage area characteristics to be READ WRITE.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.39 Additional Memory Utilized for Global Buffers Starting With Rdb Release 7.1.0.4

Bug 2904319

In Oracle Rdb Release 7.1.0.4, a change was made to increase the size of an internal global buffers related data structure. Unfortunately, the number of times this structure is allocated is based on the number of global buffers, the maximum global buffer user limit and the maximum number of users allowed in the database. Databases with large numbers of users, global buffers and per user limits could experience a unexpected potentially moderate growth in the size of the global section.

This problem has been resolved in Oracle Rdb Release 7.1.2. The size of the internal data structure is now dynamically adjusted based on the actual maximum possible number of pages per buffer. For most databases (those with a maximum number of pages per buffer of 32 or less), this change causes an effective reduction in size of the data structure to the pre−V7.1.0.4 behaviour.

### 6.1.40 Fields Added to Informational Table RDB$CACHES

The following changes have been made to the RDB$CACHES information table.

- Bit 7 of column RDB$FLAGS is set when snapshots are enabled.
- A new column RDB$SNAP_CACHE_SIZE has been added which is "Number of snapshot record slots in cache".
- A new column RDB$PHYSICAL_MEMORY has been added which is "Physical memory in bytes".

The following example shows these changes.

```
SQL> show table rdb$caches
Information for table RDB$CACHES

An information table.
Columns for table RDB$CACHES:
Column Name                     Data Type        Domain
−−−−−−−−−−−                     −−−−−−−−−        −−−−−−
...
RDB$SNAP_CACHE_SIZE             INTEGER
RDB$PHYSICAL_MEMORY             BIGINT
SQL>

SQL> select RDB$FLAGS,RDB$SNAP_CACHE_SIZE,RDB$PHYSICAL_MEMORY from RDB$CACHES;
   RDB$FLAGS   RDB$SNAP_CACHE_SIZE    RDB$PHYSICAL_MEMORY
           5                  1000                 286000
         132                333333            25727110112
2 rows selected
SQL>
```

### 6.1.41 Page Locks Not Released When LOCKING IS PAGE LEVEL

Bug 2959599

When the LOCKING IS PAGE LEVEL storage area attribute was enabled. it was possible for a process to obtain a page lock and not release it when a blocking AST was delivered by another process. Other processes would stall waiting for the page until the owning process removed the page from its buffer pool. The process could continue to hold the lock even after committing its transaction.

When this problem occurred, the output from the ***RMU /SHOW LOCKS /MODE=CULPRIT*** would be similar to the following:

```
========================================================================
SHOW LOCKS/LOCK/MODE=CULPRIT Information
========================================================================

−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
Resource: page 4443

         ProcessID Process Name    Lock ID   System ID Requested Granted
```

```
         --------- --------------- --------- --------- --------- -------
Blocker: 2541851A  DKOM_TSG_004... 5300BA60  0001002A            PW
Waiting: 25418513  DKOM_BRZ_PAS... 48001A32  0001002A  PW        NL
```

The OpenVMS system analyzer (SDA) utility SHOW LOCK command would show output similar to the following:

```
Lock id:  5300BA60            PID:     0061011A   Flags: VALBLK  CONVERT NOQUEUE
Par. id:  21010D65            SUBLCKs:        0          SYNCSTS SYSTEM  PROTECT
LKB:      FFFFFFFF.7E352E50   BLKAST: 00000000
Priority:     0000

Granted at    PW   00000000-FFFFFFFF

Resource:          0000115B 00000050  P...[...  Status: PROTCT
 Length   08       00000000 00000000  ........
 Exec. mode        00000000 00000000  ........
 System            00000000 00000000  ........

Local copy
```

Note that in the above output, the blocking AST address (BLKAST) is zero and the lock is granted in PW mode.

To avoid this problem, set the storage area to LOCKING IS ROW LEVEL.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.42 Potential Bugcheck Using VLM Global Buffers With Galaxy Support

Previously, databases using the Global Buffers VLM (Very Large Memory) feature in an OpenVMS Galaxy environment could experience a bugcheck indicating reading an invalid page number. The exception in the bugcheck dump file could look somewhat similar to the following:

```
***** Exception at 001A2124 : PIOFETCH$WITHIN_DB + 000005D4
%RDMS-F-CANTREADDBS, error reading pages 1:0-0
-RDMS-F-BADPAGNUM, page 0 is out of valid range (1:270) for physical area 1
```

This problem typically would occur with databases where a second node attempted to open the database after it had already been opened and accessed.

This problem has been corrected in Oracle Rdb Release 7.1.2. During VLM global buffer mapping in a Galaxy environment, Oracle Rdb now does not zero the contents of the global buffer pages.

## 6.1.43 Incorrect Retrieval of Duplicates from Ranked Indexes

Bugs 2903118 and 2669387

When retrieving data using an index of *TYPE IS SORTED RANKED*, where the index had duplicates, records could fail to be delivered or could be delivered more than once.

This problem was most noticeable where *HOLD* cursors were used across transactions and rows were added or deleted from the duplicate chain being retrieved. It was possible, but very much less likely, that the problem could occur where the duplicate chain being retrieved was updated by the same attach.

The problem only occurred where data was being retrieved from a duplicates chain, for example multiple records with the same key value in the index being used for retrieval.

In addition, the problem only occurred if the retrieval was reset. A reset will occur if the transaction state has changed since the last fetch or if the duplicates chain being retrieved has been updated since the last fetch. In both these cases, Oracle Rdb must re−compute its position in the duplicates chain being retrieved.

In example 1, the rows being retrieved from the cursor are being deleted after each fetch. The problem occurred because the same DBKEY was delivered twice.

```
SQL> fetch cursor_8 into :db_key;
SQL> delete from MY_TABLE
cont> where DBKEY = :db_key;
%RDB−E−NO_RECORD, access by DBKEY failed because DBKEY is
no longer associated with a record
−RDMS−F−NODBK, 49:105813:6 does not point to a data record
```

In extremely rare situations, it was possible that multiple rows could be delivered more than once or the cursor could even loop through the selected duplicates chain indefinitely.

In example 2, the switch between transactions causes a *HOLD* cursor to be reset. In this case, the DBKEY being returned is incorrectly reconstructed from the duplicates bitmap and is invalid. The DBKEY 110:81:1 does not exist in the database.

```
fetch a into :i,:j;
commit;
set trans read write;
commit;
fetch a into :i,:j;
%SQL−F−UDCURDEL, Cursor in fetch, update or delete, positioned
on a deleted record
−RDMS−F−NODBK, 110:81:1 does not point to a data record
```

In some situations, it was possible that a row that should be retrieved was not delivered. The row may simply be missing from the result or the cursor may prematurely return an end of stream status.

The problem only occurred when updates to the duplicate chain being retrieved caused the cursor to be reset. The problem only occurred where the index used for retrieval was of *TYPE IS SORTED RANKED*, and only if there were duplicates for the key value being fetched.

The problem can be avoided by adding additional columns to the index to ensure that all key values are unique.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.44 ALTER TABLE Statement May Fail if RESERVING Clause Used for Transaction

Bug 2979800

In prior releases of Oracle Rdb 7.1, the processing of a DEFAULT clause or an AUTOMATIC clause might fail if the value expression referenced a table that was not referenced in the RESERVING clause of the SET TRANSACTION statement that precedes the ALTER TABLE statement.

The following examples show the reported error when the table is referenced.

```
SQL> set transaction
cont>     read write
cont>     reserving MY_TAB2 for exclusive write;
SQL>
SQL> alter table MY_TAB2
cont>     alter column a default (select min(a) from MY_TAB1);
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-DEFINCCOL, DEFAULT is incompatible with datatype of column "MY_TAB2"."A"
-RDB-E-UNRES_REL, relation !AC in specified request is not a relation reserved in specifi
SQL>
SQL> alter table MY_TAB2
cont>     add column c automatic insert as (select avg (a) from MY_TAB1);
%RDB-E-NO_META_UPDATE, metadata update failed
-RDB-E-UNRES_REL, relation MY_TAB1 in specified request is not a relation reserved in spe
SQL>
```

These errors occur when an implicit UPDATE statement is executed which assigns the AUTOMATIC AS, AUTOMATIC INSERT AS or DEFAULT value expression to each previously inserted row.

This problem has been corrected in Oracle Rdb Release 7.1.2. Rdb now implicitly reserves these referenced tables for SHARED READ during the special UPDATE query.

## 6.1.45 Processes Not Recovered After Node Failure

Bug 2893496

It was possible for failed processes to not be recovered after a "node failure" if a recovery process were to fail before completion. For example, if there was a system crash and after the system rebooted database recovery processes (DBRs) were started but before they completed the system crashed again, the processes being recovered by the DBRs at the time of the second system crash might not be recovered again by subsequent DBRs. Those processes may or may not be recovered if a "node failure" recovery was needed in the future. If those processes had made updates to the database and the database was modified after the DBR failure, then it was possible for the database to become corrupt.

This problem has been corrected in Oracle Rdb Release 7.1.2. Database recovery failures will no longer cause failed processes to become forgotten.

## 6.1.46 DBR Bugchecks at PIO$COMPLETE + 000002E4

Bug 2968254

The database recovery (DBR) process would sometimes fail with the following exception:

```
***** Exception at 0019EB34 : PIO$COMPLETE + 000002E4
%SYSTEM−F−ACCVIO, access violation, reason mask=00,
virtual address=000000003FC68000, PC=000000000019EB34, PS=0000001B
```

The calling routine ("Saved PC") was:

```
Saved PC = 00109AE4 : KODBND$REL_AREAS_BLOCKING_ASTX + 000000D4
```

This problem would occur during a small timing window when a DBR process was running down and database maintenance activities were being done at the same time.

To avoid this problem, wait for all recovery processes to complete before proceeding with database maintenance that requires storage areas to be moved or deleted.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.47 Left Outer Join Query With CONCAT Function Returns Wrong Results

Bugs 3139961, 2836144, 1837522

The following query with CONCAT function returns wrong results (should return 4 rows).

```
set flags 'strategy,detail';
select T1.YEAR,T1.WEEK,T1.KODE
  FROM  T1 LEFT OUTER JOIN T2
          ON T1.YEAR = T2.YEAR AND T1.KODE = T2.KODE,
        T3 where
    (T1.YEAR||T1.WEEK <='200337') and
    T3.IPROC = T1.IPROC AND
    T3.ITEAM = T1.ITEAM     ;
Tables:
  0 = T1
  1 = T2
  2 = T3
Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation 2:T3
      Index name  T3_IND1 [0:0]
  Cross block entry 2
    Conjunct: ((0.YEAR < SUBSTRING ('200337' FROM 0 FOR 4)) AND
              <error: missing expression>) OR ((0.YEAR = SUBSTRING ('200337'
               FROM 0 FOR 4)) AND (0.WEEK <= SUBSTRING ('200337' FROM 4)))
    Cross block of 2 entries        (Left Outer Join)
      Cross block entry 1
        Leaf#01 FFirst 0:T1 Card=4
          Bool: (((0.YEAR < SUBSTRING ('200337' FROM 0 FOR 4)) AND NOT
                MISSING (0.WEEK)) OR ((0.YEAR = SUBSTRING ('200337'
                 FROM 0 FOR 4)) AND (0.WEEK <= SUBSTRING ('200337' FROM 4)))
```

```
                  ) AND (2.IPROC = 0.IPROC) AND (2.ITEAM = 0.ITEAM)
            BgrNdx1 T1_IND2 [0:0] Fan=15
            BgrNdx2 T1_IND1 [0:0] Fan=14
              Bool: ((0.YEAR < SUBSTRING ('200337' FROM 0 FOR 4)) AND NOT
                    MISSING (0.WEEK)) OR ((0.YEAR = SUBSTRING ('200337'
                     FROM 0 FOR 4)) AND (0.WEEK <= SUBSTRING ('200337' FROM 4)
                    ))
        Cross block entry 2
          Conjunct: (0.YEAR = 1.YEAR) AND (0.KODE = 1.KODE)
          Index only retrieval of relation 1:T2
            Index name  T2_IND1 [2:2]      Direct lookup
              Keys: (0.YEAR = 1.YEAR) AND (0.KODE = 1.KODE)
 T1.YEAR   T1.WEEK   T1.KODE
 2003      26        270
 2003      34        270
2 rows selected
```

The problem is caused by the fix made for Bug 1837522 in Oracle Rdb Release 7.0.6.2 where a left outer join query with OR predicate returns wrong results.

Even though this query apparently does not contain an OR predicate, the Oracle Rdb optimizer transforms the CONCAT function into an OR expression with two SUBSTRING functions, as seen in the following detail strategy.

```
  Bool: (((0.YEAR < SUBSTRING ('200337' FROM 0 FOR 4)) AND
          NOT MISSING (0.WEEK)) OR
         ((0.YEAR = SUBSTRING ('200337' FROM 0 FOR 4)) AND
          (0.WEEK <= SUBSTRING ('200337' FROM 4)))) AND
        (2.IPROC = 0.IPROC) AND (2.ITEAM = 0.ITEAM)
```

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.1.48 RCS Bugchecks at DIOCCH$UNMARK_GRIC_ENT

Bug 2502144

In prior releases of Oracle Rdb, it was possible for the Record Cache Server (RCS) process to bugcheck in DIOCCH$UNMARK_GRIC_ENT. This problem was due to an incorrect check in the RCS process while evaluating the amount of locked space on a database page that is owned by the RCS.

When the RCS process writes an erased or resized record back to the database page, it must return the resultant locked space on the page back to free space. This is because the RCS is not allowed to "own" locked space. As part of this action, the RCS was checking to make sure that the amount of locked space being returned exactly matched the length of the space being returned by the erased or resized record. This check was not taking into account the possibility that there was existing locked space on the page marked with the TID of the RCS. In such a case, the RCS could find that it "owned" more locked space than it expected and would bugcheck.

This problem has been corrected in Oracle Rdb Release 7.1.2. The RCS process now correctly evaluates and releases all locked space on the page that is owned by the RCS after writing an erased or resized record back to the database.

## 6.1.49 Query With Sum Function of Two Select Counts Bugchecks

Bug 2649215

A query with a sum function of two select counts could produce a bugcheck.

```
    select sum ((select count (*) - (select count (*) from T1
                                      where T1.F1 = T2.F1
                                        and T1.F2 = T2.F2)
             from T2
             where F3 = 'B'
             group by T2.F1, T2.F2)
             )
    from rdb$database;
%DEBUG-I-DYNMODSET, setting module RDMS$PREEXEASN
%SYSTEM-F-BREAK, breakpoint fault at PC=003995E9, PSL=03C00004
break on exception at RDMS$PREEXEASN\RDMS$$FIND_VALID_SEG_CRTV\%LINE 8443 in
EAD 1
```

The query works if one of the equality predicates is removed, as in the following example.

```
select sum ((select count (*)
                - (select count (*) from T1 where
                     T1.F1 = T2.F1
!---                 and T1.F2 = T2.F2
                    )
             from T2 where F3 = 'B'
             group by T2.F1, T2.F2
             ))
from rdb$database;
Tables:
  0 = RDB$DATABASE
  1 = T2
  2 = T1
Aggregate: 0:SUM (<agg1>)
Cross block of 2 entries
  Cross block entry 1
    Aggregate: 1:VIA (<mapped field> - <agg2>)
    Cross block of 2 entries
      Cross block entry 1
        Aggregate: 2:COUNT (*)
        Index only retrieval of relation 2:T1
          Index name  U1_T1 [1:1]
            Keys: 2.F1 = 1.F1
      Cross block entry 2
        Aggregate: 3:COUNT (*)
        Conjunct: 1.F3 = 'B'
        Get     Retrieval by index of relation 1:T2
          Index name  U1_T2 [0:0]
  Cross block entry 2
    Retrieval sequentially of relation 0:RDB$DATABASE

                    1
1 row selected
```

The query also works if the SUM function is removed.

```
select count (*) - (select count (*) from T1
                                  where T1.F1 = T2.F1
                                    and T1.F2 = T2.F2)
             from T2
             where F3 = 'B'
             group by T2.F1, T2.F2
             ;
Tables:
  0 = T2
  1 = T1
Cross block of 2 entries
  Cross block entry 1
    Aggregate: 0:COUNT (*)
    Conjunct: 0.F3 = 'B'
    Get     Retrieval by index of relation 0:T2
      Index name  U1_T2 [0:0]
  Cross block entry 2
    Aggregate: 1:COUNT (*)
    Index only retrieval of relation 1:T1
      Index name  U1_T1 [2:2]        Direct lookup
        Keys: (1.F1 = 0.F1) AND (1.F2 = 0.F2)

                    0
1 row selected
```

There is no known workaround for this problem.

This problem has been corrected in Oracle Rdb Release 7.1.2.

# 6.2 SQL Errors Fixed

## 6.2.1 DECLARE LOCAL TEMPORARY TABLE Limited to 10 Tables Per Session

Bug 2911428

In prior releases of Oracle Rdb, access to tables declared using the DECLARE LOCAL TEMPORARY TABLE statement in interactive and dynamic SQL would fail. This problem does not occur when DECLARE LOCAL TEMPORARY TABLE is used in a CREATE MODULE statement.

The following example shows the reported error.

```
SQL> select * from demo.MODULE.prodn_new_constraints_table;
%RDB-E-OBSOLETE_METADA, request references metadata objects that no longer exist
-RDMS-F-TABIDNOTDEF, relation ID, 11, is not defined in database
```

This problem has been corrected in Oracle Rdb Release 7.1.2. This limitation has been removed from interactive and dynamic SQL. A workaround would be to declare tables #10 and #11 and not use those temporary tables.

## 6.2.2 Unexpected ACCVIO When Reporting Incompatible Character Set Assignments

Bug 3098432

In prior releases of Rdb, an incompatible character set assignment might cause Rdb to generate a malformed message vector which can lead to an ACCVIO while trying to display the messsage.

The following example shows the problem.

```
SQL> select count(*) from rdb$database
cont>  where _dec_mcs'a'=translate('a' using rdb$dec_kanji);
%RDB-E, invalid or unsupported data conversion
%SYSTEM-F-ACCVIO, access violation, reason mask=00, virtual address=
00000000004F2DE4, PC=FFFFFFFF80207624, PS=0000001B
```

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.2.3 ALTER STORAGE MAP May Fail With PARTEXTS Error for LIST Storage Map

Bug 3030789

In prior releases of Oracle Rdb V7.1, a failure could occur when the ALTER STORAGE MAP was used to change the LIST storage map for the database.

The following example uses MF_PERSONNEL to show this problem.

```
SQL> create storage map lists_map
cont> store list
cont>     in JOBS for (resumes)
cont>     in MF_PERS_SEGSTR;
SQL>
SQL> alter storage map lists_map
cont> store list
cont>     in JOBS for (resumes)
cont>     in (EMPIDS_LOW, EMPIDS_MID, MF_PERS_SEGSTR) for (employees)
cont>     in MF_PERS_SEGSTR;
%RDB-E-NO_META_UPDATE, metadata update failed
-RDMS-E-PARTEXTS, partition "SYS_P00001" already exists for this map or index
"LISTS_MAP"
```

The LIST storage maps which exhibit this problem use the same storage area more than once. In such cases, the partition should be given a unique name and this is done by CREATE STORAGE MAP but not by ALTER STORAGE MAP. The name "SYS_P00001" in the error is a name generated by Rdb.

This problem has been corrected in Oracle Rdb Release 7.1.2.

A workaround to this problem is to include a partition name in the map and so avoid the implicit naming performed by Rdb.

```
SQL> alter storage map lists_map
cont> store list
cont>     in JOBS (partition a1) for (resumes)
cont>     in (EMPIDS_LOW (partition b1)
cont>        ,EMPIDS_MID (partition b2)
cont>        ,MF_PERS_SEGSTR (partition b3)) for (employees)
cont>     in MF_PERS_SEGSTR (partition c1);
SQL>
```

## 6.2.4 Problems Corrected in ALTER INDEX ... BUILD PARTITION

Bug 3069318

Several problems with the ALTER INDEX partitioning building facility have been corrected with this release of Rdb.

1. Use of the SIZE IS clause on an index segment could cause a bugcheck dump on either a SORTED or SORTED RANKED index.
   The bugcheck summary for a SORTED RANKED index would look similar to this output:

   ```
   COSI-F-BUGCHECK, internal consistency failure
   Exception occurred at PSIIBUILD2BUILDFROMBOTTOM + 000017F4
   Called from PSII2CREATETREE + 000002E8
   Called from RDMS$$KOD_CREATE_TREE + 000001E4
   ```

   The bugcheck summary for a SORTED index would look similar to this output:

   ```
   COSI-F-BUGCHECK, internal consistency failure
   ```

```
Exception occurred at PSIIBUILD$BUILD_FROM_BOTTOM + 00001354
Called from PSII$CREATE_TREE + 0000018C
Called from RDMS$$KOD_CREATE_TREE + 000002B8
```

This problem occurred because the SIZE IS restriction was not applied during the SORT of the partition rows. Thus rows were sorted by the full key and this could cause the data to be returned in the incorrect collating order.

2. The index segment prefix cardinality was not calculated correctly. All columns were stored with the same value. The RMU Collection Optimizer_Statistics command should be executed on all affected indices.
3. BUILD PARTITION would fail if executed on a UNIQUE HASHED index.
   The bugcheck summary would look similar to this output:

```
SYSTEM-F-ACCVIO, access violation
Exception occurred at symbol not found
Called from symbol not found
Called from RDMS$$CHANGE_INDEX + 000020AC
Called from RDMS$$RELEASE_DDL_VM_HNDLR + 00001BA4
```

   The problem was that cardinality collection is not done for the index or the index segments when the index is a HASHED index. However, the BUILD command still attempted to access and store the accumulated cardinality data.

These problems have been corrected in Oracle Rdb Release 7.1.2.

## 6.2.5 INSERT Into Table With an IDENTITY Column May Fail With RDB$_NO_PRIV Error

Bug 3051390

The following example shows the reported error when an unprivileged user attempts to implicitly reference the IDENTITY sequence via the INSERT statement. The SHOW PRIVILEGE statement shows that the user has no access to the sequence even though a SHOW PROTECTION command shows the required access has been granted to the user.

```
SQL> INSERT INTO IDENTITY_TEST
cont> (FIELD2)
cont> VALUE
cont> ('B01');
%RDB-E-NO_PRIV, privilege denied by database facility
SQL> SHOW PRIVILEGE ON SEQUENCE IDENTITY_TEST;
Privileges on Sequence IDENTITY_TEST
    (IDENTIFIER=[DOC,NONPRIV_USER],ACCESS=NONE)
SQL> SHOW PROTECTION ON SEQUENCE IDENTITY_TEST;
Protection on Sequence IDENTITY_TEST
    (IDENTIFIER=[DOC,NONPRIV_USER],ACCESS=SELECT+REFERENCES)
    (IDENTIFIER=[DOC,SYS_DBA],ACCESS=SELECT+SHOW+ALTER+DROP+DBCTRL+REFERENCES)
    (IDENTIFIER=[*,*],ACCESS=NONE)
```

SHOW PROTECTION shows the access control list (ACL) assigned to the object. SHOW PRIVILEGE shows the access granted to the current user after processing the ACL, applying databases roles (or rights identifiers) and overriding OpenVMS privileges. However, in this case, Rdb is erroneously inheriting no access for the sequence.

The problem occurs because the CREATE TABLE command was incorrectly setting the SYSTEM flag for column−identity sequences and this prevents them from being executed from an INSERT statement.

This problem has been corrected in Oracle Rdb Release 7.1.2. The SYSTEM flag is no longer set for identity sequences.

After installing Oracle Rdb Release 7.1.2, any ALTER SEQUENCE or COMMENT ON SEQUENCE command can be executed on the identity sequence to clear the incorrectly set SYSTEM flag.

## 6.2.6 IMPORT May Generate ACCVIO Exception During Import of a Module

In previous releases of Oracle Rdb, the IMPORT command would fail with an ACCVIO exception during import of a module that contained an external function or procedure.

The following example shows the exception.

```
SQL> import database
cont>    from SQL_CREATE_MODULE_4G_EXP
cont>    filename SQL_CREATE_MODULE_4G_DB
cont> ;
%SYSTEM−F−ACCVIO, access violation, reason mask=00,
virtual address=00000000, PC=0029908E, PSL=03C00001
```

This problem has been corrected in Oracle Rdb Release 7.1.2. SQL IMPORT now correctly handles these types of modules.

## 6.2.7 Unexpected Failure From DROP STORAGE AREA ... CASCADE Clause

Bug 2814690

In prior releases of Oracle Rdb, the DROP STORAGE AREA ... CASCADE clause may raise an exception while trying to evaluate constraints for a table which has been implicitly deleted.

The following example shows this on a slightly modified MF_PERSONNEL database.

```
SQL> alter database filename MF_PERSONNEL
cont>  drop storage area EMPIDS_LOW cascade
cont>  drop storage area EMPIDS_MID cascade
cont>  drop storage area EMPIDS_OVER cascade;
%RDB−E−EXT_ERR, Oracle Rdb extension error
−RDMS−F−RELNEXTS, relation EMPLOYEES does not exist in this database
```

Each DROP STORAGE AREA ... CASCADE clause checks to see if this is the last partition for the table and, if so, it implicitly drops the table. Unfortunately, that table name was already queued for constraint verification.

This problem has been corrected in Oracle Rdb Release 7.1.2. The table is now removed from the constraint verification list when it is implicitly dropped.

A workaround is to execute each DROP STORAGE AREA ... CASCADE clause in a separate ALTER DATABASE statement.

```
SQL> alter database filename MF_PERSONNEL
cont>  drop storage area EMPIDS_LOW cascade;
SQL> alter database filename MF_PERSONNEL
cont>  drop storage area EMPIDS_MID cascade;
SQL> alter database filename MF_PERSONNEL
cont>  drop storage area EMPIDS_OVER cascade;
```

## 6.2.8 Create Module Declaring Integer Variable's Default With Cast Bugchecks

Bug 2672904

The problem occurs when processing a CREATE MODULE statement that has a global variable with a default specified that involves a CAST and which contains a literal value. Such a statement would fail and generate a SQL bugcheck dump.

The following example shows a query which fails due to this condition.

```
SQL> at 'fi personnel';
SQL> create module bug_test
cont> declare :A_VALUE integer = cast(ABS(-100) as integer)
cont> Function TEST() return integer;
cont> begin
cont> return :A_VALUE;
cont> end;
cont> end module;
%RDMS-I-BUGCHKDMP, generating bugcheck dump file
RDB_USER10:[HOWARD]SQLBUGCHK.DMP;
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=0000000000000000, PC=00000000004E2EEC, PS=0000001B
```

The SQLBUGCHK.DMP for the above example has an entry similar to the following:

```
***** Exception at 004E2EEC : SQL$$BLR_MODULE_VAR_GEN + 0000029C
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=0000000000000000, PC=00000000004E2EEC, PS=0000001B
```

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.2.9 Incorrect Unit for the DETECTED ASYNC PREFETCH THRESHOLD Option

Bug 2838771

In prior versions of Oracle Rdb V7.1, the THRESHOLD option of the DETECTED ASYNC

PREFETCH clause required the units to be PAGES. However, this value is really specified in BUFFERS.

To avoid confusion, the SQL syntax has now been changed to use the BUFFERS unit for this clause. The older syntax is now deprecated as shown in the following example:

```
SQL> alter database
cont>     filename 'DB$:PERSONNEL'
cont>     detected async prefetch is ENABLED
cont>         (depth is 4 buffers, threshold is 4 pages);
%SQL−I−DEPR_FEATURE, Deprecated Feature:
PAGES is replaced with BUFFERS
```

This problem has been corrected in Oracle Rdb Release 7.1.2. In addition, the RMU Extract command will output the new corrected syntax.

```
SQL> alter database
cont>     filename 'DB$:PERSONNEL'
cont>     detected async prefetch is ENABLED
cont>         (depth is 4 buffers, threshold is 4 buffers);
```

## 6.2.10 IVP or Other Failure With Dynamic SQL if SQL$INT is Installed /RESIDENT

Bug 2950983

In SYS$STARTUP:SQL$STARTUP.COM, if the line RESIDENT = "/RESIDENT" was present, (for example: not commented out), the IVP failed while running the dynamic SQL test.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.2.11 CREATE INDEX Would Fail With READ_ONLY_FIELD Error

Bug 2979800

In prior releases of Oracle Rdb V7.1, it was not possible to create a partitioned index which used an IDENTITY, AUTOMATIC AS or an AUTOMATIC INSERT AS column as one of the partitioning columns.

The following example shows the error that was generated.

```
SQL> create sequence TESTING_SEQUENCE;
SQL>
SQL> create table TESTING_TABLE
cont>     (a automatic as TESTING_SEQUENCE.nextval);
SQL>
SQL> create index TESTING_INDEX
cont>     on TESTING_TABLE (a)
cont>     store using (a)
cont>         in TEST2A with limit of (1000)
```

```
cont>         in TEST2B with limit of (2000)
cont> ;
%RDB−E−NO_META_UPDATE, metadata update failed
−RDB−E−READ_ONLY_FIELD, attempt to update the read−only field A
SQL>
```

The same error is generated by the ALTER INDEX statement if an non−partitioned index is changed to a partitioned index.

The problem occurs when an implicit INSERT ... PLACEMENT ONLY RETURNING statement is used and assigns values to these read−only columns. This PLACEMENT ONLY statement doesn't actually insert a row but allows the return of DBKEY and other information related to the proposed index.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.2.12 SQL Added Padding Spaces to Saved Source SQL Statement

Bugs 1570063 and 2424124

The various SQL ALTER and CREATE statements save the original statement source in the database for display by SQL SHOW commands and RMU Extract /Item=MODULE. In previous versions, padding spaces were added to the saved source lines which made the output from SHOW and RMU Extract hard to read.

With this release of Rdb, SQL no longer adds these padding spaces to the saved sources. This means that the SHOW commands will no longer add any indentation when displaying most sources (triggers, constraints, storage maps, and modules).

Objects created with older versions of Rdb will still contain the padding spaces.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.2.13 ALTER TABLE Caused AUTOMATIC UPDATE Columns to be Evaluated for All Rows

Bug 2897401

In prior releases of Oracle Rdb V7.1, some forms of ALTER TABLE would erroneously cause AUTOMATIC UPDATE columns to be evaluated for all rows in the table. This occurred because these statements execute an implicit UPDATE statement on the table.

- ALTER TABLE ... ADD COLUMN ... DEFAULT ...
  ALTER TABLE ... ADD COLUMN ... AUTOMATIC ...
  When a new column was added that included the DEFAULT attribute, then all previously inserted rows would be updated to include the DEFAULT as the value of the new column.
  If the new column was based on a domain which included the DEFAULT attribute then, in the same fashion, the table would be updated.
  If the new column was an AUTOMATIC INSERT AS or an AUTOMATIC AS (implying

both insert and update actions), then all previously inserted rows would be updated with the AUTOMATIC expression as the value of the new column.
- ALTER TABLE ... DROP COLUMN ...
  When a LIST OF BYTE VARYING column is dropped, the table is implicitly updated to remove the list data. This is done because the data is stored separately from the table rows and Rdb must collect all references to those lists.

This problem has been corrected in Oracle Rdb Release 7.1.2. These implicit UPDATE statements no longer cause the AUTOMATIC UPDATE columns to be evaluated.

***Related Changes***

The AUTO_OVERRIDE flag can be used to allow updates to selected AUTOMATIC columns during INSERT so that rows could be reloaded, or during UPDATE to adjust incorrectly stored values. The effect of this flag has changed slightly with this release of Oracle Rdb.

- For the INSERT statement, 'AUTO_OVERRIDE' allows assignment to any AUTOMATIC column and any insert AUTOMATIC column omitted from the column list will be evaluated normally. This remains the same as in previous versions.
- For the UPDATE statement, 'AUTO_OVERRIDE' allows direct assignment of values to any AUTOMATIC column. No AUTOMATIC columns are evaluated.
  The UPDATE case has changed with this release of Rdb. In prior releases, any UPDATE, even with AUTO_OVERRIDE set, would cause any unassigned automatic columns to be evaluated. This had a side effect for some DDL operations which implicitly performed updates on the base table.

To accommodate applications that wish to retain the older behavior, the DEFAULT clause can be used to assign the AUTOMATIC expression to any column requiring complete reprocessing. If the DEFAULT clause is used in an INSERT or UPDATE statement, then one of the following will be applied:

- If a DEFAULT attribute is present for the column, then that value will be applied during INSERT or UPDATE.
- Else if an AUTOMATIC attribute is present for the column, then that value will be applied during INSERT or UPDATE. This can only happen if the SET FLAGS 'AUTO_OVERRIDE' is used since during normal processing these columns are read−only.
- Otherwise a NULL will be applied during INSERT or UPDATE.

## 6.2.14 SQL−F−NODBFILE When SQL Modules are Compiled With /CONNECT

Bug 3002999

When multiple SQL modules were called from the same main program and a disconnect was done, under some circumstances calls to SQL modules after the disconnect would fail with the following message:

```
%SQL−F−NODBFILE, Alias <ALIAS_NAME> is missing a declaration
```

Where <ALIAS_NAME> is one of the aliases declared by one of the SQL modules. The behavior will occur whenever the following is true:

- The SQL modules are compiled with /CONNECT (which is the default).
- Some aliases have a run time resolution for the filename or pathname.
- One or more modules do not declare one of the aliases which has a run−time resolution.
- The first call after the disconnect is to a module which does not have a declaration of an alias with a run−time resolution.

As a workaround, ensure that the first SQL module called after the disconnect declares all aliases which have a run−time resolution.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.2.15 Concatenate Now Supports Non−character Values in ORACLE LEVEL2

Bug 2948230

The following examples show that concatenating non−character string values result in an error from interactive SQL.

```
SQL> create table t2(i1 int, c1 char(3));
SQL> select i1||c1 from t2;
%SQL−F−UNSNUMXPR, Unsupported numeric expression
```

This release of Oracle Rdb now supports this functionality for compatiblity with Oracle RDBMS. You must use the dialect ORACLE LEVEL2 to enable this feature.

```
SQL> set dialect 'oracle level2';
SQL> create table t2(i1 int, c1 char(3));
SQL> insert into t2 values (1,'a');
1 row inserted
SQL> insert into t2 values (12,'ab');
1 row inserted
SQL> insert into t2 values (123,'abc');
1 row inserted
SQL> select i1||c1 from t2;

 1a
 12ab
 123abc
3 rows selected
```

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.2.16 RDB−E−REQ_NO_TRANS With Multiple SQL Modules and Images

Bugs 2770532 and 1808821

When two SQL Modules are used in the same process from different images using a shared connection, SQL mixes the state of the two modules. For example, suppose there is a main program which directly calls some SQL Module Language routines. Further, this program calls routines that reside in a shared image which in turn call other SQL Module Language routines. (It doesn't matter whether different databases are used but it does matter that the SQL Modules are compiled to use a single connection − i.e. /NOCONNECT which is the default.). In this configuration, the second and subsequent calls to SQL Modules in either image will return a SQLCODE of −1. The underlying error will be reported as:

```
%RDB−E−REQ_NO_TRANS, attempt to execute request with no transaction active
```

As a workaround, SQL Modules can be compiled with /CONNECT. Note: this workaround has two potential drawbacks:

- The application may encounter the problem documented in Bug 3002999.
- If the application depends on a single connection (e.g. in order to have a shared transaction context), it will not work as designed.

This problem has been corrected in Oracle Rdb Release 7.1.2.

# 6.3 RDO and RDML Errors Fixed

## 6.3.1 RDML /DATE_TYPE Qualifier Default is Now NOEMPTY_RECORDS

RDML /PASCAL generates a record type for date items. This type is either an empty record or a record composed of two longword elements depending on the value of the /DATE_TYPE qualifier (EMPTY_RECORDS or NOEMPTY_RECORDS respectively). Earlier versions of the Pascal compiler behaved as desired when empty records were used in assignments and fetches. More current versions of the Pascal compiler give the following warning message when empty records are used for dates:

```
%PASCAL−W−EMPTYVAR, Fetching an empty record with an explicit size
attribute may not yield expected results
```

Furthermore, statements flagged with the above warning frequently do not yield the desired result but instead treat the record as having a zero length. Using the /DATE_TYPE=NOEMPTY_RECORD qualifier avoids this problem.

In order to make RDML /PASCAL more usable, NOEMPTY_RECORDs has been made the default value for the DATE_TYPE. Additionally, using an explicit /DATE_TYPE=EMPTY_RECORDS on the RDML /PASCAL command line will now result in the following warning message being issued by RDML:

```
%RDML−W−DATE_NOEMPTY, /DATE_TYPE=EMPTY_RECORDS specified;
use /DATE_TYPE=NOEMPTY_RECORDS
```

# 6.4 RMU Errors Fixed

## 6.4.1 Could Not Import Statistics on Different Node

In prior releases of Oracle Rdb, it was not possible to take a statistics file created via the ***RMU/CLOSE/STATISTICS=EXPORT*** and import it on another node. For example, if a database was backed up and then restored on another system the ***RMU/OPEN/STATISTICS=IMPORT*** command would ignore the statistics file (.RDS file) even if it was renamed to contain the nodename of the new node.

This problem has been corrected in Oracle Rdb Release 7.1.2. The statistics file may now be imported on a node that is not the node that wrote the file.

## 6.4.2 RMU Extract Generates Incorrect ALTER TABLE ... ADD CONSTRAINT Syntax

Bug 3109136

In prior versions of Oracle Rdb, RMU Extract could generate incorrect ALTER TABLE ... ADD CONSTRAINT syntax when the constraint contained a reference to the DBKEY for the table.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.4.3 RMU/VERIFY/CONSTRAINTS Problems With Named Tables And Constraints

Bug 3016789

Oracle Rdb V7.0 RMU/VERIFY/CONSTRAINTS verified all constraints even if a list of constraints was specified. Oracle Rdb V7.1 RMU/VERIFY/CONSTRAINTS verified all constraints for all tables even if a list of tables was specified. This behaviour has been corrected so that only constraints for a specified list of tables or only the constraints specified will be verified.

The following example shows that for Oracle Rdb V7.0 RMU/VERIFY/CONSTRAINTS, all constraints were verified even if a list of constraints was specified and that for Oracle Rdb V7.1 RMU/VERIFY/CONSTRAINTS, all constraints for all tables were verified even if a list of tables was specified.

```
$DEFINE RDMS$DEBUG_FLAGS "H"
$ rmu/show version
Executing RMU for Oracle Rdb V7.0−7
$ rmu/verify/constraint=(constraint=degrees_foreign2)/log mf_personnel
%RMU−I−BGNROOVER, beginning root verification
%RMU−I−ENDROOVER, completed root verification
%RMU−I−BGNVCONST, beginning verification of constraints for database
device:[directory]MF_PERSONNEL.RDB;1
~H Extension (VERIFY CONSTRAINTS) Item List: (len=0)
~H: ...verify constraint "WORK_STATUS_PRIMARY_STATUS_CODE"
~H: ...verify constraint "STATUS_NAME_VALUES"
```

```
~H: ...verify constraint "STATUS_TYPE_VALUES"
~H: ...verify constraint "EMPLOYEES_PRIMARY_EMPLOYEE_ID"
~H: ...verify constraint "EMP_SEX_VALUES"
~H: ...verify constraint "EMP_STATUS_CODE_VALUES"
~H: ...verify constraint "JOBS_PRIMARY_JOB_CODE"
~H: ...verify constraint "WAGE_CLASS_VALUES"
~H: ...verify constraint "DEPARTMENTS_PRIMARY1"
~H: ...verify constraint "JOB_HISTORY_FOREIGN1"
~H: ...verify constraint "JOB_HISTORY_FOREIGN2"
~H: ...verify constraint "JOB_HISTORY_FOREIGN3"
~H: ...verify constraint "SALARY_HISTORY_FOREIGN1"
~H: ...verify constraint "COLLEGES_PRIMARY_COLLEGE_CODE"
~H: ...verify constraint "DEGREES_FOREIGN1"
~H: ...verify constraint "DEGREES_FOREIGN2"
~H: ...verify constraint "DEG_DEGREE_VALUES"
~H: ...verify constraint "CANDIDATES_LAST_NAME_NOT_NULL"
~H: ...verify constraint "RESUMES_UNIQUE_EMPLOYEE_ID"
~H: ...verify constraint "RESUMES_FOREIGN1"
%RMU−I−ENDVCONST, completed verification of constraints for database
device:[directory]MF_PERSONNEL.RDB;1

$ rmu/show version
Executing RMU for Oracle Rdb V7.1−10
$ rmu/verify/constraint=(table=colleges)/log mf_personnel
%RMU−I−BGNROOVER, beginning root verification
%RMU−I−ENDROOVER, completed root verification
%RMU−I−BGNVCONST, beginning verification of constraints for database
device:[directory]MF_PERSONNEL.RDB;1
~H Extension (VERIFY CONSTRAINTS) Item List: (len=0)
~H: ...verify constraint "WORK_STATUS_PRIMARY_STATUS_CODE"
~H: ...verify constraint "STATUS_NAME_VALUES"
~H: ...verify constraint "STATUS_TYPE_VALUES"
~H: ...verify constraint "EMPLOYEES_PRIMARY_EMPLOYEE_ID"
~H: ...verify constraint "EMP_SEX_VALUES"
~H: ...verify constraint "EMP_STATUS_CODE_VALUES"
~H: ...verify constraint "JOBS_PRIMARY_JOB_CODE"
~H: ...verify constraint "WAGE_CLASS_VALUES"
~H: ...verify constraint "DEPARTMENTS_PRIMARY1"
~H: ...verify constraint "JOB_HISTORY_FOREIGN3"
~H: ...verify constraint "JOB_HISTORY_FOREIGN1"
~H: ...verify constraint "JOB_HISTORY_FOREIGN2"
~H: ...verify constraint "COLLEGES_PRIMARY_COLLEGE_CODE"
~H: ...verify constraint "DEGREES_FOREIGN1"
~H: ...verify constraint "DEGREES_FOREIGN2"
~H: ...verify constraint "DEG_DEGREE_VALUES"
~H: ...verify constraint "CANDIDATES_LAST_NAME_NOT_NULL"
~H: ...verify constraint "RESUMES_UNIQUE_EMPLOYEE_ID"
~H: ...verify constraint "RESUMES_FOREIGN1"
%RMU−I−ENDVCONST, completed verification of constraints for database
device:[directory]MF_PERSONNEL.RDB;1
```

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.4.4 RMU/BACKUP/PARALLEL/DISK_FILE Did Not Work Properly

The Oracle Rdb RMU/BACKUP/PARALLEL/DISK_FILE command, which backs up a database to multiple disk .RBF files using multiple processes, did not work properly. It created an empty directory list in the PLAN file and access violated even if the directory list was edited into the PLAN file and the PLAN file was then executed. These problems have been corrected and parallel backup to multiple disk files now creates a valid PLAN file and executes properly.

The following example shows that a parallel backup to multiple disk files created a PLAN file with empty disk directory lists.

```
$ RMU /BACKUP -
    /PARALLEL=EXECUTOR_COUNT=3 -
    /DISK_FILE=(WRITER_THREADS=1) -
    /LIST_PLAN=MFP.PLAN -
    /NOEXECUTE -
    MF_PERSONNEL -
    DISK1:[DIRECTORY]MFP,DISK2:[DIRECTORY],DISK3:[DIRECTORY]
$TYPE MFP.PLAN

! Plan created on  6-JUN-2003 by RMU/BACKUP.

Plan Name = MFP
Plan Type = BACKUP

Plan Parameters:
    Database Root File = disk:[directory]MF_PERSONNEL.RDB;1
    Backup File = MFP.RBF
    Style = Multifile

End Plan Parameters

Executor Parameters :
    Executor Name = COORDINATOR
    Executor Type = Coordinator
End Executor Parameters

Executor Parameters :
    Executor Name = WORKER_001
    Executor Type = Worker
    ! Executor Node = Node name for executor
    Start Storage Area List
        MF_PERS_SEGSTR,
        EMPIDS_MID,
        JOBS
    End Storage Area List
    Writer_threads = 1
    Directory List
    End Directory List
End Executor Parameters

Executor Parameters :
    Executor Name = WORKER_002
    Executor Type = Worker
    ! Executor Node = Node name for executor
    Start Storage Area List
        DEPARTMENTS,
        EMPIDS_OVER,
        SALARY_HISTORY
    End Storage Area List
    Writer_threads = 1
    Directory List
    End Directory List
End Executor Parameters
```

```
Executor Parameters :
    Executor Name = WORKER_003
    Executor Type = Worker
    ! Executor Node = Node name for executor
    Start Storage Area List
        EMPIDS_LOW,
        EMP_INFO,
        RDB$SYSTEM
    End Storage Area List
    Writer_threads = 1
    Directory List
    End Directory List
End Executor Parameters
```

The following example shows that a parallel backup to multiple disk files now creates a PLAN file with disk directory lists that contain the disk directories from the command line.

```
$ RMU /BACKUP -
    /PARALLEL=EXECUTOR_COUNT=3 -
    /DISK_FILE=(WRITER_THREADS=1) -
    /LIST_PLAN=MFP.PLAN -
    /NOEXECUTE -
    MF_PERSONNEL -
    DISK1:[DIRECTORY]MFP,DISK2:[DIRECTORY],DISK3:[DIRECTORY]
$RMU/BACKUP/PLAN MFP.PLAN
$TYPE MFP.PLAN

! Plan created on  6-JUN-2003 by RMU/BACKUP.

Plan Name = MFP
Plan Type = BACKUP

Plan Parameters:
    Database Root File = disk:[directory]MF_PERSONNEL.RDB;1
    Backup File = MFP.RBF
    Style = Multifile

End Plan Parameters

Executor Parameters :
    Executor Name = COORDINATOR
    Executor Type = Coordinator
End Executor Parameters

Executor Parameters :
    Executor Name = WORKER_001
    Executor Type = Worker
    ! Executor Node = Node name for executor
    Start Storage Area List
        MF_PERS_SEGSTR,
        EMPIDS_MID,
        JOBS
    End Storage Area List
    Writer_threads = 1
    Directory List
    DISK1:[DIRECTORY]
    End Directory List
End Executor Parameters
```

```
Executor Parameters :
    Executor Name = WORKER_002
    Executor Type = Worker
    ! Executor Node = Node name for executor
    Start Storage Area List
        DEPARTMENTS,
        EMPIDS_OVER,
        SALARY_HISTORY
    End Storage Area List
    Writer_threads = 1
    Directory List
    DISK2:[DIRECTORY]
    End Directory List
End Executor Parameters

Executor Parameters :
    Executor Name = WORKER_003
    Executor Type = Worker
    ! Executor Node = Node name for executor
    Start Storage Area List
        EMPIDS_LOW,
        EMP_INFO,
        RDB$SYSTEM
    End Storage Area List
    Writer_threads = 1
    Directory List
    DISK3:[DIRECTORY]
    End Directory List
End Executor Parameters
```

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.4.5 RMU/BACKUP/DISK_FILE Fails if too Many Writer Threads

Bug 2991564

The Oracle Rdb RMU/BACKUP/DISK_FILE command, which backs up a database to multiple disk files, could fail with an access violation error if the /WRITER_THREADS parameter specified a number of writer threads that exceeded the number of disk devices listed on the command line. For RMU/BACKUP/DISK_FILE, there cannot be more writer threads than disk devices specified on the command line. Therefore, Oracle Rdb RMU has been changed so that if the specified WRITER_THREADS value is larger than the number of disk devices, a warning message will be output and the number of writer threads will be changed to equal the number of disk devices. The message output is:

```
%RMU−W−DEFWRITER, The specified WRITER THREADS value is too large −
changing to maximum possible value of #
```

The following example shows the access violation that occurred if the number of writer threads (in this case 5) exceeded the number of disk devices (in this case 3).

```
$RMU/BACKUP/DISK=(WRITER_THREADS=5) MF_PERSONNEL DISK1:MFP.RBF,DISK2:,DISK3:
%RMU−I−RESUME, resuming operation on volume 2 using _disk2
%RMU−I−RESUME, resuming operation on volume 3 using _disk3
%SYSTEM−F−ACCVIO, access violation, reason mask=00,
```

```
virtual address =0000000000000030, PC=00000000003CA174, PS=0000001B
%RMU−F−FATALERR, fatal error on BACKUP
%RMU−F−FTL_BCK, Fatal error for BACKUP operation at  4−JUN−2003 05:59:00.11
```

The following example shows the warning message that now is output to show that the number of writer threads has been changed to equal the number of disk devices.

```
$RMU/BACKUP/DISK=(WRITER_THREADS=5) MF_PERSONNEL DISK1:MFP.RBF,DISK2:,DISK3:
%RMU−W−DEFWRITER, The specified WRITER THREADS value is too large − changing
to maximum possible value of 3
%RMU−I−RESUME, resuming operation on volume 2 using _disk2
%RMU−I−RESUME, resuming operation on volume 3 using _disk3
```

The workaround for this problem is to specify a number of writer threads which is equal to or less than the number of disk devices.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.4.6 TRUNCATE TABLE and RMU /REPAIR Corruption Corrected

Previously, when using the TRUNCATE TABLE statement followed by a RMU /REPAIR /ABM command executed one or more times, various corruptions of the SPAM and ABM structures could occur. This could sometimes lead to table records that had been truncated "reappearing" in the table unexpectedly.

The following example demonstrates one possible symptom of this problem detected by RMU /VERIFY:

```
$ SQL$
    CREATE DATABASE FILENAME 'TEST.RDB'
        CREATE STORAGE AREA RDB$SYSTEM FILENAME 'RDB$SYSTEM.RDA'
        CREATE STORAGE AREA AREA FILENAME 'AREA.RDA';
    CREATE TABLE TAB (ID INTEGER);
    CREATE INDEX IND ON TAB (ID) STORE IN AREA;
    CREATE STORAGE MAP TM FOR TAB STORE IN AREA;
    COMMIT;
    INSERT INTO TAB VALUES (1);
1 row inserted
    COMMIT;
    TRUNCATE TABLE TAB;
    COMMIT;
    EXIT;
$!
$ RMU/VER/ALL/NOLOG TEST
%RMU−I−NODATANDX, no data records in index IND
$!
$ RMU/REPAIR/ABM TEST
%RMU−I−FULBACREQ, A full backup of this database should be
performed after RMU REPAIR
$ RMU/VER/ALL/NOLOG TEST
%RMU−W−AIPLAREID, area inventory page 152 entry #9 contains a
                  reference to logical area 58 that is nonexistent
%RMU−W−BADABMPTR, invalid larea for ABM page 5 in storage area 2.
                  The SPAM page entry for this page is for a different larea.
                  SPAM larea_dbid : 0 page larea_dbid: 58.
```

```
%RMU-W-BADABMPTR, invalid larea for ABM page 6 in storage area 2.
                  The SPAM page entry for this page is for a different larea.
                  SPAM larea_dbid : 0 page larea_dbid: 58.
%RMU-W-BADABMPTR, invalid larea for ABM page 7 in storage area 2.
                  The SPAM page entry for this page is for a different larea.
                  SPAM larea_dbid : 0 page larea_dbid: 58.
%RMU-E-BADABMPAG,       error verifying ABM pages
%RMU-I-NODATANDX, no data records in index IND
```

This problem has been corrected in Oracle Rdb Release 7.1.2. The RMU /REPAIR /ABM command no longer incorrectly updates the SPAM and ABM structures after a TRUNCATE TABLE operation.

## 6.4.7 AIJ Backup File May Not Allow Recovery of a Restored Database

Bug 2515818

When using circular AIJ files, the following sequence of commands will end up with an AIJ backup file (foo$2.aij) looking like the second example.

Example 1

```
$ rmu/backup foo [bck]foo.rbf
$ rmu/backup/after foo [.bck]foo$0.aij
$ sql$ drop database filename [.db]foo;
$ rmu/restore/nocdd [.bck]foo.rbf
$ rmu/recover [.bck]foo$0
$ rmu/backup/quiet foo [.bck]foo2.rbf
$!
$! Work with database : insert/update/delete records
$!
$ rmu/backup/after/quiet foo [.bck]foo$2.aij
```

Example 2

```
1/1              TYPE=O (open record with :)
    AIJ Sequence Number is 2
    Last Commit TSN is 0:192
2/2              TYPE=K (close record)
129/3            TYPE=O (open record with :)
    AIJ Sequence Number is 3
    Last Commit TSN is 0:224
             <some other AIJ records>
132/10           TYPE=K (final close record)
```

Note that we have a gap in the TSN numbers between the first and second Open records (192 vs 224). In this particular case, the AIJ backup file (foo$2.aij) will not allow you to recover a database restored from your last backup (foo2.rbf above), as shown in the following log excerpt :

```
$ rmu/restore/norecover/nocdd/directory=[.db2] [.bck]foo2.rbf
$ rmu/recover/log/trace/root=[.db2]foo [.bck]foo$2.aij
 :
%RMU-W-AIJSEQPRI, AIJ file sequence number 2 created prior to expected
sequence 3
%RMU-I-AIJONEDONE, AIJ file sequence 2 roll-forward operations
completed
```

```
%RMU-I-LOGRECSTAT, transaction with TSN 0:224 ignored
%RMU-I-LOGRECSTAT, transaction with TSN 0:256 ignored
%RMU-I-LOGRECSTAT, transaction with TSN 0:288 ignored
%RMU-I-AIJONEDONE, AIJ file sequence 3 roll-forward operations completed
%RMU-I-LOGRECOVR, 0 transactions committed
```

As a possible workaround, start the recovery with the AIJ backup file used to recover the database after it has been dropped (foo$0.aij in the above example).

```
$ RMU/RECOVER/LOG/TRACE/ROOT=[.DB2]FOO [.BCK]FOO$0.AIJ,[.BCK]FOO$2.AIJ
```

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.4.8 Domains Required for SQL_FUNCTION Not Output by RMU Extract

Bug 1684510

In prior releases, the special domains created by the SQL_FUNCTIONS.SQL script were not extracted by RMU Extract. This meant that any database that had the SQL functions library installed would not be correctly recreated using the output from RMU Extract.

The following example shows the error that is generated.

```
%SQL-F-NO_SUCH_FIELD, Domain RDB$ORACLE_SQLFUNC_VCHAR_DOM
does not exist in this database or schema
```

This problem has been corrected in Oracle Rdb Release 7.1.2. The following domains are explicitly extracted if present in the source database:

- RDB$ORACLE_SQLFUNC_CHAR_DOM CHAR(1)
- RDB$ORACLE_SQLFUNC_DATE_DOM DATE VMS
- RDB$ORACLE_SQLFUNC_DEC_MCS_DOM CHAR(1)
- RDB$ORACLE_SQLFUNC_VCHAR_DOM VARCHAR(2000)

The workaround is to include these definitions in the generated script manually or automate it using a DCL procedure.

## 6.4.9 RMU /RESTORE Bugchecks at LCK$STALL_FOR_ENQ + 0CC0

It was possible for ***RMU /RESTORE*** to bugcheck with the following stack trace when the ***/JUST_PAGE*** or ***/ONLINE*** qualifiers were used:

```
***** Exception at 00661910 : LCK$STALL_FOR_ENQ + 00000CC0
%SYSTEM-F-ACCVIO, access violation, reason mask=00,
virtual address=0000000000000048, PC=0000000000661910, PS=0000001B

Saved PC = 0065F630 : LCK$LOCK + 000006F0
Saved PC = 00408048 : RMUDIO$ACQUIRE_PAGE_LOCKS + 00000528
Saved PC = 003AF458 : RMUCLI$RESTORE + 00005038
```

To avoid the problem, ensure that no other users are accessing the database while the restore operation is in progress.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.4.10 Changes to RMU /CLOSE Behavior

Bug 2490013

Oracle Rdb Release 7.1.2 introduces changes in behavior to the ***RMU /CLOSE*** command.

In prior releases, if a database was not open on the current node and the ***RMU /CLOSE*** command was issued with the /WAIT qualifier, a RDMS−F−DBNOTACTIVE error was returned. In this release, the RDMS−F−DBNOTACTIVE error is no longer returned if the database is not open and the /WAIT qualifier is specified.

This change was made to accommodate the ***/CLUSTER*** operation. When /WAIT is specified, it implies ***/CLUSTER***. When ***/CLUSTER*** is in effect, the database must be opened on the local node before the shutdown request can be relayed to other nodes that may have the database open. Thus, using the /WAIT qualifier causes the database to be opened on the local node if it isn't already open. Since /WAIT causes the database to be opened, then there is no RDMS−F−DBNOTACTIVE error returned.

Similarly, in prior releases, if a database was not open on the current node and the ***/CLUSTER*** and /NOWAIT qualifiers were specified, then an error was returned and the database was not closed on other nodes. In this release, no error is returned and the database is closed on other nodes.

Note that if opening the database on the node issuing the ***RMU /OPEN*** command will cause the number of allowed cluster nodes to be exceeded, then an error will be returned and the database will not be closed on the other nodes.

## 6.4.11 May Not be Able to Apply the Next AIJ After Doing RMU /RECOVER /RESOLVE /STATE Multiple Times

Bug 2494572

In a distributed transaction with the prepare record in one AIJ and the commit record in a second AIJ, if you ran RMU /RECOVER /RESOLVE /STATE twice on the first AIJ, then the recovery of the second AIJ would fail as shown below. This is because an incorrect AIJ sequence number had been set up during the second recovery of the first journal.

```
$ RMU /RECOVER /RESOLVE /STATE=ABORT /ROOT=FOODB AIJ_1.AIJ
%RMU−I−LOGRECDB, recovering database file FOODB.RDB;1
%RMU−I−AIJONEDONE, AIJ file sequence 0 roll−forward operations completed
_AIJ_file:
%RMU−I−AIJALLDONE, after−image journal roll−forward operations completed
%RMU−I−AIJFNLSEQ, to start another AIJ file recovery, the sequence number needed will be
%RMU−I−AIJNOENABLED, after−image journaling has not yet been enabled

$ RMU /RECOVER /RESOLVE /STATE=ABORT /ROOT=FOODB AIJ_1.AIJ
%RMU−I−LOGRECDB, recovering database file FOODB.RDB;1
```

```
%RMU-W-AIJSEQPRI, AIJ file sequence number 0 created prior to expected sequence 1
%RMU-I-AIJONEDONE, AIJ file sequence 0 roll-forward operations completed
%RMU-W-NOTRANAPP, no transactions in this journal were applied
_AIJ_file:
%RMU-I-AIJALLDONE, after-image journal roll-forward operations completed
%RMU-I-AIJFNLSEQ, to start another AIJ file recovery, the sequence
number needed will be 0 <=
%RMU-I-AIJNOENABLED, after-image journaling has not yet been enabled

$ RMU /RECOVER /RESOLVE /STATE=ABORT /ROOT=FOODB AIJ_2.AIJ
%RMU-I-LOGRECDB, recovering database file FOODB.RDB;1
%RMU-F-AIJNORCVR, recovery of this journal must start with sequence 0
%RMU-F-FTL_RCV, Fatal error for RECOVER operation at  5-DEC-2002 09:39:51.73
```

As a potential workaround, recover all the journals in the same recovery command.

```
$ RMU /RECOVER /RESOLVE /STATE=ABORT /ROOT=FOODB AIJ_1.AIJ,AIJ_2.AIJ
```

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.4.12 RMU /EXTRACT /ITEM=DATABASE May Not Display Snapshot File Attributes

Bug 2713879

The following example shows the unexpected output from RMU Extract /Item=DATABASE for the snapshot attributes of a new storage area.

```
   .
   .
   .
    create storage area AREA2
        filename 'USER1:[TESTING]AREA2.RDA'
        -- read write storage area
        locking is row level
        page format is UNIFORM
        page size is 2 blocks
        allocation is 600 pages
        extent is (minimum 99, maximum 9999, percent growth 20)
        snapshot allocation is 0 pages
        snapshot extent is (minimum 0, maximum 0, percent growth 0)
        snapshot checksum calculation is DISABLED
   .
   .
   .
```

This problem occurs when the database is opened on more than one node in a cluster and is altered to add one or more storage areas. RMU Extract /Item=DATABASE and /Item=IMPORT require that the file information be refreshed on the node before the attributes are fetched. This was not being done for the snapshot attributes.

This problem has been corrected in Oracle Rdb Release 7.1.2. Oracle Rdb now automatically refreshes the snapshot file information prior to returning information to RMU Extract.

## 6.4.13 Recovery of Empty Optimized AIJ Does Not Update the Sequence Number

Bug 2581948

The recovery of an empty optimized AIJ does not update the next AIJ sequence number. This will prevent the recovery of the next AIJ as shown below. An optimized AIJ file can be empty if the corresponding AIJ file contains only rolled−back transactions. See the following example.

```
$ sql$
SQL> attach 'filename opt_aij';
SQL> insert into t1 values (1);
1 row inserted
SQL> rollback;
SQL> exit
$ RMU /BACKUP /AFTER OPT_AIJ OPT_AIJ_BCK1

− Note that opt_aij_bck1 contains a single rolled−back transaction

$ sql$
SQL> attach 'filename opt_aij';
SQL> insert into t1 values (2);
1 row inserted
SQL> commit;
SQL> exit
$ RMU /BACKUP /AFTER OPT_AIJ OPT_AIJ_BCK2

! opt_aij_bck2 contains a single committed transaction

$ RMU /OPTIMIZE /AFTER OPT_AIJ_BCK1 OPT_AIJ_OPT1
$ RMU /OPTIMIZE /AFTER OPT_AIJ_BCK2 OPT_AIJ_OPT2
$ RMU /RESTORE /NOCDD OPT_AIJ

! Recovering the empty Optimized AIJ file opt_aij_opt1

$ RMU /RECOVER /LOG OPT_AIJ_OPT1.OAIJ
%RMU−I−LOGRECDB, recovering database file $111$DUA4:[VIGIER.OPTAIJ.DB]OPT_AIJ.RDB;1
%RMU−I−LOGOPNAIJ, opened journal file RAID1:[VIGIER.OPTAIJ.DB]OPT_AIJ_OPT1.OAIJ;1 at  8−
%RMU−I−AIJONEDONE, AIJ file sequence 0 roll−forward operations completed
%RMU−I−LOGRECOVR, 0 transactions committed
%RMU−I−LOGRECOVR, 0 transactions rolled back
%RMU−I−LOGRECOVR, 0 transactions ignored
%RMU−I−AIJNOACTIVE, there are no active transactions
%RMU−I−AIJSUCCES, database recovery completed successfully
%RMU−I−AIJNXTSEQ, to continue this AIJ file recovery, the sequence number needed will be
%RMU−I−AIJALLDONE, after−image journal roll−forward operations completed
%RMU−W−NOTRANAPP, no transactions in this journal were applied
%RMU−I−AIJSUCCES, database recovery completed successfully
%RMU−I−AIJFNLSEQ, to start another AIJ file recovery, the sequence number needed will be
%RMU−I−AIJNOENABLED, after−image journaling has not yet been enabled

− Note that the next AIJ Sequence number has been left as 0

− Recovering the second Optimized AIJ file opt_aij_opt2
$ RMU /RECOVER /LOG OPT_AIJ_OPT2.OAIJ
%RMU−I−LOGRECDB, recovering database file $111$DUA4:[VIGIER.OPTAIJ.DB]OPT_AIJ.RDB;1
%RMU−F−AIJNORCVR, recovery of this journal must start with sequence 0
%RMU−F−FTL_RCV, Fatal error for RECOVER operation at  8−JAN−2003 10:23:38.32
```

The recovery fails since it does not have the expected AIJ sequence number.

A workaround is to put all the optimized AIJ files in the same command line.

```
$ RMU /RECOVER /LOG OPT_AIJ_OPT1.OAIJ,OPT_AIJ_OPT2.OAIJ
```

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.4.14 RMU /VERIFY Not Writing Constraint Verification Failure Warnings to /OUTPUT File

Bug 2813243

RMU /VERIFY put CONSTFAIL warnings to the terminal but didn't log the message in the /OUTPUT file.

This problem has been corrected in Oracle Rdb Release 7.1.2.

```
$ RMU /VERIFY /CONSTRAINTS /LOG /OUTPUT=M.OUT -
    /TRANSACTION_TYPE=READ MF_PERSONNEL
%RMU-W-CONSTFAIL, Verification of constraint "EMP_SEX_VALUES" has failed.
$ TYPE M.OUT
%RMU-I-BGNROOVER, beginning root verification
%RMU-I-ENDROOVER, completed root verification
%RMU-I-BGNVCONST, beginning verification of constraints for database
 DISK:[DIRECTORY]MF_PERSONNEL.RDB;1
%RMU-W-CONSTFAIL, Verification of constraint "EMP_SEX_VALUES" has failed.
%RMU-I-ENDVCONST, completed verification of constraints for database
 DISK:[DIRECTORY]MF_PERSONNEL.RDB;1
%RMU-I-DBBOUND,   bound to database "SQL_USER2:[NIKADE.TEMP]MF_PERSONNEL.RDB;1"
%RMU-I-OPENAREA,  opened storage area RDB$SYSTEM for read_only retrieval
%RMU-I-BGNAIPVER, beginning AIP pages verification
%RMU-I-ENDAIPVER, completed AIP pages verification
%RMU-I-BGNABMSPM, beginning ABM pages verification
%RMU-I-ENDABMSPM, completed ABM pages verification
%RMU-I-CLOSAREAS, releasing read_only retrieval lock on all storage areas
%RMU-S-ENDVERIFY, elapsed time for verification :    0 00:00:06.16
```

## 6.4.15 RMU /COLLECT May Default to a READ WRITE Transaction

Bug 2898115

When no transaction type is specified for the *RMU/COLLECT OPTIMIZER_STATISTICS* command, RMU examines the database storage areas and if any storage area has snapshots disabled, a read write transaction is used. If no storage areas have snapshots disabled, then a read only transaction is used.

Currently Rdb does not allow different areas on the same database to have snapshots enabled or disabled. However, the attribute is recorded in the database root file against each storage area.

In previous versions, if a database had reserved and unused storage area slots, this check could erroneously examine the unused storage area slots and conclude that areas on the database had snapshots disabled. This could cause RMU to adopt a read write transaction as the default transaction mode for *RMU/COLLECT*.

The following example shows *RMU/COLLECT* executing against a database with reserved storage areas erroneously selecting a read write transaction. The example also shows the use of debug flags to display the transaction modes used by the command.

```
$ DEFINE RDMS$SET_FLAGS TRANSACTION
$ RMU /COLLECT OPTIMIZER MF_PERSONNEL
~T Compile transaction (1) on db: 1
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_WRITE (read write)
~T Start_transaction (1) on db: 1, db count=1
~T Commit_transaction on db: 1
~T Prepare_transaction on db: 1
~T Compile transaction (1) on db: 2
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_READ (read only)
~T Start_transaction (1) on db: 2, db count=1
~T Commit_transaction on db: 2
~T Prepare_transaction on db: 2
~T Compile transaction (1) on db: 3
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_WRITE (read write)
~T Start_transaction (1) on db: 3, db count=1
~T Commit_transaction on db: 3
~T Prepare_transaction on db: 3
~T Compile transaction (1) on db: 4
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_WRITE (read write)
~T Start_transaction (1) on db: 4, db count=1
~T Commit_transaction on db: 4
~T Prepare_transaction on db: 4
```

The following example shows the corrected behaviour.

```
$ RMU /COLLECT OPTIMIZER MF_PERSONNEL /TRANS=READ_ONLY
~T Compile transaction (1) on db: 1
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_READ (read only)
~T Start_transaction (1) on db: 1, db count=1
~T Commit_transaction on db: 1
~T Prepare_transaction on db: 1
~T Compile transaction (1) on db: 2
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_READ (read only)
~T Start_transaction (1) on db: 2, db count=1
~T Commit_transaction on db: 2
~T Prepare_transaction on db: 2
~T Compile transaction (1) on db: 3
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
```

```
0001 (00001) TPB$K_READ (read only)
~T Start_transaction (1) on db: 3, db count=1
~T Commit_transaction on db: 3
~T Prepare_transaction on db: 3
~T Compile transaction (1) on db: 4
~T Transaction Parameter Block: (len=2)
0000 (00000) TPB$K_VERSION = 1
0001 (00001) TPB$K_WRITE (read write)
~T Start_transaction (1) on db: 4, db count=1
~T Commit_transaction on db: 4
~T Prepare_transaction on db: 4
```

Note that the read write transaction at the end of the example is used to update information in the metadata and is normal and required.

Specifying the transaction type on the command line by using the */TRANSACTION_TYPE=READ_ONLY* qualifier will avoid this problem.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.4.16 RMU /CONVERT Failed if VMS$MEM_RESIDENT_USER Rights Identifier Not Held

Bug 2879451

The Oracle Rdb RMU /CONVERT of an Rdb database to V7.1 failed if the user was not granted the VMS VMS$MEM_RESIDENT_USER rights identifier and the database contained row caches defined with the attributes SHARED MEMORY IS SYSTEM and LARGE MEMORY IS ENABLED. This problem has been corrected. The conversion of a database will now succeed even if the user does not hold the VMS VMS$MEM_RESIDENT_USER rights identifier. Because of the nature of RMU /CONVERT where a database already containing the memory parameters mentioned above is being converted to V7.1 but not used, RMU /CONVERT of databases to V7.1 will now be allowed even if the user does not hold the VMS$MEM_RESIDENT_USER rights identifier. However, once the database with these memory parameters is converted to V7.1, the user will still have to hold the VMS$MEM_RESIDENT_USER rights identifier to open it.

The following example shows the failure of the conversion of an Oracle Rdb database to V7.1 that occurred if the user was not granted the VMS VMS$MEM_RESIDENT_USER rights identifier and the database contained row caches defined with the attributes SHARED MEMORY IS SYSTEM and LARGE MEMORY IS ENABLED.

```
$RMU /CONVERT TEST_DATABASE.RDB
%RMU−I−LOGCONVRT, database root converted to current structure level
%RMU−F−MONFLRMSG, failure message received from the monitor
−COSI−F−NOMEMRESID, requires rights identifier VMS$MEM_RESIDENT_USER
%RMU−F−FTL_CNV, Fatal error for CONVERT operation at  1−APR−2003 15:28:26.09
```

The workaround for this problem is to grant the user the VMS VMS$MEM_RESIDENT_USER rights identifier before executing the RMU /CONVERT of the database to V7.1.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.4.17 Multi−Disk File Restore Could Fail if READER_THREADS Exceeded One

The Oracle Rdb RMU MULTI DISK FILE RESTORE feature that allows the restore of an Oracle Rdb database from multiple backup files ("RBF" files) on disk when the /DISK_FILE qualifier is specified could cause corruption of the database when it was restored. This corruption was usually associated with the RMU−E−INVBLKHDR or the RMU−F−BACKFILCOR and RMU−E−BACFILCOR_03 errors being returned on the restore. This corruption did not always happen but could occur frequently if the READER_THREADS parameter specified a value greater than one. If a value of "1" was specified (the default if the READER_THREADS parameter is not used with the /DISK_FILE qualifier), the problem did not happen since it only could occur if multiple reader threads were executing. Note that this problem only occurred if the "/DISK_FILE" qualifier was used. There was no problem for tape and single disk file backups and restores where the /DISK_FILE qualifier is not used.

The following example shows two cases where database corruption could occur when using the /DISK_FILE qualifier and specifying a READER_THREADS value greater than "1" (the default).

```
$ RMU /BACKUP /NOLOG /JOURNAL=B31A /DISK_FILE=(WRITER_THREADS=1) MF_PERSONNEL −
        DISK:[DIRECTORY]B31A.RBF,DISK:[DIRECTORY],DISK:[DIRECTORY]

$ RMU /RESTORE /NOLOG /NOCDD /JOURNAL=B31A /DISK_FILE=(READER_THREADS=2) −
        DISK:[DIRECTORY]B31A.RBF,DISK:[DIRECTORY],DISK:[DIRECTORY]
%RMU−E−INVBLKHDR, invalid block header in backup file

$ RMU /RESTORE /NOLOG /NOCDD /JOURNAL=B31A /DISK_FILE=(READER_THREADS=2) −
        DISK:[DIRECTORY]B31A.RBF,DISK:[DIRECTORY],DISK:[DIRECTORY]
%RMU−F−BACFILCOR, Backup file is corrupt
−RMU−E−BACFILCOR_03, Unexpected condition after end of volume detected
%RMU−F−FATALERR, fatal error on RESTORE
%RMU−F−FTL_RSTR, Fatal error for RESTORE operation at 29−APR−2003 17:41:48.82
```

The workaround for this problem is to not specify a value of READER_THREADS which is greater than one. If READER_THREADS is not specified it will default to one.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.4.18 RMU Extract Not Extracting Modules Correctly When MATCH Option Used

Bug 3139946

In prior releases of Oracle Rdb V7.1, the RMU Extract Option=MATCH was applied to all routines contained within a module being extracted. This was incorrect. The MATCH qualifier should have only been applied to the primary object name and not the nested definition. The result of this error was that only routines matching the same pattern as the module name were being extracted.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.4.19 Recovery of Database With Fixed AIJs After Convert to V7.1 Could Lose Data

The recovery of a database with multiple fixed AIJ files could lose data with the following sequence. (Please also see the example below.)

1. Conversion of the V7.0 database to V7.1.
2. Switchovers of the existing fixed AIJ files until getting the message "%RMU−W−AIJMODSWTCH, AIJ switch−over suspended − add new journal or backup current".
3. A backup of the current AIJ file.
4. A restore of the backed up database.
5. A recover of the AIJ data.

The following example shows the incorrect behavior where transaction data was lost.

```
$ @sys$library:rdb$setver 7.0
$ sql
  attach 'file mf_personnel';
  drop table test_tbl1;
  create table test_tbl1 (col1 integer, col2 varchar(20));
  insert into test_tbl1 values (1, 'test only');
  commit;
  insert into test_tbl1 select col1+1, col2 from test_tbl1;
  commit;
  exit;
$ rmu/set after/reserve=4 mf_personnel
$ rmu/set after/disable mf_personnel
$ rmu/set after/add=(name=aij1,file=aij_journal1.aij) mf_personnel
$ rmu/set after/add=(name=aij2,file=aij_journal2.aij) mf_personnel
$ rmu/set after/add=(name=aij3,file=aij_journal3.aij) mf_personnel
$ rmu/set after/add=(name=aij4,file=aij_journal4.aij) mf_personnel
$ rmu/set after/enable mf_personnel
$ rmu/back/after mf_personnel aij.bck1
$ rmu/backup mf_personnel personnel.rbf_1
$ sql
  attach 'file mf_personnel';
  select * from test_tbl1;
        COL1   COL2
           1   test only
           2   test only
  exit
$ @sys$library:rdb$setver 7.1
$ rmu/convert/commit/noconfirm mf_personnel
$ rmu/backup mf_personnel personnel.rbf_2
$ sql
 attach 'file mf_personnel';
 update test_tbl1 set col1=col1+60;
 commit;
 exit
$ rmu/set after/switch mf_personnel
$ sql
 attach 'file mf_personnel';
 update test_tbl1 set col2='test 1' where col1 < 62;
 commit;
 exit
$ rmu/set after/switch mf_personnel
$ sql
```

```
 attach 'file mf_personnel';
 insert into test_tbl1 select * from test_tbl1;
 commit;
 select * from test_tbl1;
        COL1   COL2
          61   test 1
          62   test only
          61   test 1
          62   test only
 exit
$ rmu/set after/switch mf_personnel
%RMU−W−AIJMODSWTCH, AIJ switch−over suspended − add new journal or backup
 current
$ rmu/back/after mf_personnel aij.bck2
$ del *.rdb;
$ del *.rda;
$ del *.snp;
$ rmu/restore/nocdd personnel.rbf_2
$ rmu/recover aij.bck2
$ sql
 attach 'file mf_personnel';
 select * from test_tbl1;
        COL1  COL2
        61    test 1
        62    test only
 exit;
```

The following example shows the correct results at the end of the sequence with all transaction data preserved.

```
$ sql
 attach 'file mf_personnel';
 select * from test_tbl1;
        COL1  COL2
        61    test 1
        62    test only
        61    test 1
        62    test only
        exit;
```

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.4.20 Some Database Backup Files Created With LZSS Compression Could Not be Restored

Bug 3113361

Some database backup files created with /COMPRESSION or /COMPRESSION=LZSS specified on the backup command for using the LZSS method of software compression (the default) could not be restored. This did not happen if /COMPRESSION=HUFFMAN had been specified on the backup for using the HUFFMAN method of software compression. On the restore, an access violation occurred and the restore did not complete resulting in a corrupt, partly restored database.

This did not happen for smaller root files such as the root file for the MF_PERSONNEL database but did happen for larger root files with a large number of storage areas. In our testing, this problem happened for root files which reserved a number of storage areas equal to 64 or greater.

This problem has now been fixed. Specifying /COMPRESSION or /COMPRESSION=LZSS on the backup command will now cause no problems on the restore. Note that the /COMPRESSION qualifier can only be used with the RMU/BACKUP command since the RMU/RESTORE command determines whether a backup file has been compressed from information saved in the backup file.

The following example shows the previous failing behavior. If the database was backed up by specifying /COMPRESSION or /COMPRESSION=LZSS, the backup file was created without error but an access violation could occur on the restore.

```
$ rmu/backup/nolog/compress test.rdb test.rbf
$ sql drop database file test;
$ rmu/restore/nocdd/log test.rbf
%RMU-I-RESTXT_00, Restored root file device:[directory]TEST.RDB;1
%RMU-I-RESTXT_21, Starting full restore of storage area (RDB$SYSTEM)
 device:[directory]TEST_SYSTEM.RDA;1 at 25-AUG-2003 13:30:13.32
%RMU-I-RESTXT_21, Starting full restore of storage area (TEST_TABLES)
 device:[directory]TEST_TABLES.RDA;1 at 25-AUG-2003 13:30:13.32
%SYSTEM-F-ACCVIO, access violation, reason mask=04, virtual
 address=0000000000F0C678, PC=FFFFFFFF810E9B10, PS=0000001B
%RMU-I-BUGCHKDMP, generating bugcheck dump file device:[directory]RMUBUGCHK.DMP;
%RMU-F-FATALERR, fatal error on RESTORE
%RMU-F-FTL_RSTR, Fatal error for RESTORE operation at 25-AUG-2003 13:30:13.71
```

To avoid this problem, specify RMU/BACKUP/COMPRESSION=HUFFMAN or do not use the /COMPRESSION qualifier on the backup command.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.4.21 RMU Verify Access Violation Allocating Memory for Expanding a Storage Record

Bug 3079474

An access violation sometimes occurred in RMU/VERIFY when insufficient memory was allocated to create a buffer in virtual memory for expanding storage records. This caused the verify to terminate abnormally. This would only occur if database corruption was present in the database. This problem has been fixed.

The following example shows the access violation during the RMU/VERIFY.

```
$ rmu/ver/all/nolog

%RMU-E-ERRSEGFET, Error fetching segmented string's primary segment.
%RMU-I-SEGSTRDBK, Segmented string is at logical dbkey 1:10757:2.
%RMU-I-SEGRECDBK, Data record is at logical dbkey 7:33099:0.
%RMU-W-BADSTAREA, invalid storage area DBID 42941,
                  valid storage areas are between 1 and 1222
%RMU-E-RDYSEGSTR, ready needed for segmented string at 1:10533:17
%RMU-E-ERRSEGFET, Error fetching segmented string's primary segment.
%RMU-I-SEGSTRDBK, Segmented string is at logical dbkey 1:10533:17.
%RMU-I-SEGRECDBK, Data record is at logical dbkey 7:33098:1.
%RMU-F-ABORTVER,  fatal error encountered; aborting verification
%SYSTEM-F-ACCVIO, access violation, reason mask=04, virtual
address=000000000EA8DAC0, PC=0000000000311F34, PS=0000001B
%RMU-F-FATALOSI, Fatal error from the Operating System Interface.
```

```
%RMU−I−BUGCHKDMP, generating bugcheck dump file DISK:[DIRECTORY]RMUBUGCHK.DMP;
%RMU−F−FTL_VER, Fatal error for VERIFY operation at 21−AUG−2003 00:38:05.60
```

This problem has been corrected in Oracle Rdb Release 7.1.2.

# 6.5 LogMiner Errors Fixed

## 6.5.1 LogMiner Elimination of Processing Unneeded AIJ Files

By default, after−image journal files are processed in the order that they are presented to the RMU UNLOAD AFTER_JOURNAL command. The ORDER_AIJ_FILES qualifier specifies that the input after−image journal files are to be processed in ascending order by sequence number. This can be of benefit when you use wildcard (* or %) processing of a number of input files. The .AIJ files are each opened, the first block is read (to determine the sequence number), and the files are closed prior to the sorting operation.

The /RESTART=restart−point qualifier can be used to specify an AIJ Extract Restart Control Point (AERCP) that indicates the location to begin the extraction. The AERCP indicates the transaction sequence number (TSN) of the last extracted transaction along with a location in the .AIJ file where a known "Micro−quiet point" exists.

When the Restart qualifier is not specified and no input after−image journal files are specified on the command line, the Continuous LogMiner process starts extracting at the beginning of the earliest modified online after−image journal file.

Previously, all specified input after−image journal files were always processed. This behaviour has been altered when both the RESTART and ORDER_AIJ_FILES qualifiers are specified. In this situation, any after−image journal files containing only sequence numbers prior to the "Micro−quiet point" sequence number (indicated in the AIJ Extract Restart Control Point (AERCP)) can be eliminated from processing after the order of sequence numbers has been determined by the sort operation. Eliminating the unneeded after−image journal files can speed the restart operation.

## 6.5.2 Replication Option and LogMiner Features Active at the Same Time

Bug 2903092

When both the Replication Option and LogMiner(TM) features are enabled, it is possible for the "pre−delete" record contents stored in the after−image journal file for the LogMiner to contain incorrect contents. This problem may be indicated by errors from the RMU /UNLOAD /AFTER_JOURNAL command such as the following example:

```
%RMU−W−RECVERDIF, Record at DBKEY 819:20615:8 in table
 "FOOBAR" version 262 does not match current version.
```

This problem was caused by an incorrect buffer being used when writing the "pre−delete" record contents for the LogMiner. This buffer was also used by the Replication Option code path and the existing saved content was lost.

This problem has been corrected in Oracle Rdb Release 7.1.2. A different buffer is used to save "pre−delete" record content data.

### 6.5.3 RMU /UNLOAD /AFTER_JOURNAL Created .RRD Content Clarification

Bug 2916639

An optional "RECORD_DEFINITION" keyword can be used with the RMU /UNLOAD /AFTER_JOURNAL command to create a template .RRD (record definition) file that can be used to load a transaction table. The TABLE_DEFINITION keyword can also be used to create a template SQL procedure to create such a transaction table.

The behaviour of the RMU /UNLOAD /AFTER_JOURNAL command when used with these keywords is to append the string "RDB_LM_" to the table name to create the record name in the .RRD or .SQL file.

However, when the existing table name exceeds 24 characters in length, the resultant name for the transaction table in the .SQL or .RRD file exceeds 31 characters and is no longer a valid table name in an Rdb database. In these cases, a decision about the table name to be used must be made and the .RRD or .SQL file must be manually modified.

Oracle Rdb engineering is considering alternatives for future releases to help reduce the impact of this behaviour.

### 6.5.4 /TRANSACTION_TYPE Qualifier for RMU /UNLOAD /AFTER_JOURNAL

The RMU /UNLOAD /AFTER_JOURNAL command would start either a read−write or a read−only transaction to read the database metadata. A read−only transaction would be started if the database was set to "SNAPSHOTS ARE IMMEDIATE"; otherwise a read−write transaction would be started.

The qualifier "/TRANSACTION_TYPE=" has been added to the RMU /UNLOAD /AFTER_JOURNAL command to allow explicit control over the transaction type used when reading the database metadata.

The following keywords may be specified to the "/TRANSACTION_TYPE=" qualifier.

- AUTOMATIC − The transaction type will depend upon the current database settings for snapshots (enabled, deferred, or disabled), transaction modes available to this user, and the standby status of this database.
- READ_ONLY − Starts a READ ONLY transaction.
- WRITE − Starts a READ WRITE transaction.
- ISOLATION_LEVEL − The transaction isolation level. It accepts the following keywords:
    - READ_COMMITTED
    - REPEATABLE_READ
    - SERIALIZABLE

  Please refer to the Oracle Rdb7 SQL Reference Manual under the SET TRANSACTION statement for a complete description of these isolation levels.
- WAIT − Will wait indefinitely on a locked resource.

- ♦ WAIT=n − This instructs Rdb to wait 'n' seconds before aborting the wait and the RMU session. Specifying a wait timeout interval of zero (0) is equivalent to specifying NOWAIT.
- ♦ NOWAIT − Will not wait on locked resources.

If the qualifier /TRANSACTION_TYPE is omitted or specified with no options, then the default is /TRANSACTION_TYPE=(AUTOMATIC, WAIT).

Although a WRITE transaction is started on the database, RMU /UNLOAD /AFTER_JOURNAL does not attempt to write to the database tables.

# 6.6 Row Cache Errors Fixed

## 6.6.1 Storage Area Grows Despite Row Erasure When Using Row Cache

Normally, when a process erases a row in the database, it keeps track of the page number of the deleted row to be used as a starting point for future inserts into the table. This helps processes reuse space in the database. Previously, however, when cached rows were erased, the process would not correctly maintain this starting point page number.

This problem has been reduced in scope in Oracle Rdb Release 7.1.2. When a process erases a cached row, if the database page for the row is in memory and is locked for update, the row is erased in the cache and on the database page and the starting page number for future inserts is updated. If the database page is not locked for update, however, the process does not update the database page after it erases the row in cache. The RCS (Row Cache Server) process will, at some point in the future, move the erased row back to disk. Until that time, the space on the database page remains allocated and unavailable for reuse.

For tables with heavy delete activity when snapshots are enabled in cache, it may be wise to configure the cache to specify that the RCS is to checkpoint it to the database rather than the backing store files. This will cause space to be reclaimable by moving erased rows back to the database more rapidly.

Note that applications that attach to the database for long periods and insert and erase rows in such a table may still need to periodically detach from the database and reattach in order to be able to find space for reuse to insert records. This is due to the RCS being unable, at present, to notify processes that it has moved erased rows from cache to disk and thus potentially making space available for inserting records. Oracle expects to address this behaviour in a future Oracle Rdb release.

## 6.6.2 Logical Area Record Erasure Count Not Updated for Cached Rows

Bug 3099718

Previously, logical area statistics for record erase operations were not correctly counted when erasing rows in a row cache.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.6.3 Commit Performance Improvement With Row Cache Feature

Performance of COMMIT operations has been improved when the Row Cache feature is enabled for a database. Previously, all working set entries in all caches for the user were evaluated to determine if there were any modified records that needed to be written back to the cache when the transaction committed. This scanning is now avoided when there are known to be no modified records that need to be written back.

### 6.6.4 RMU /SET ROW_CACHE Command Updates

Bug 2618405

The "RMU /SET ROW_CACHE" command now allows the Row Cache "Backing Store" directory specification to be maintained on a per−database and per−cache basis.

The "RMU /SET ROW_CACHE /BACKING_STORE_LOCATION=devdir" command can be used to specify the database default backing store location for all caches that do not have a cache−specific backing store location. The "RMU /SET ROW_CACHE /NOBACKING_STORE_LOCATION" qualifier can be used to remove the database default backing store location.

New keywords "BACKING_STORE_LOCATION=devdir" and "NOBACKING_STORE_LOCATION" have been added to the /ALTER qualifier to allow the backing store location directory to be specified on a per−cache basis.

The "NAME=cachename" keyword now accepts the wildcard characters asterisk (*) and percent sign (%) to specify matching cache names.

The "ENABLE" and "DISABLE" keywords have been removed and new keywords "DROP" and "SNAPSHOT_SLOT_COUNT" have been added.

The valid keywords for the ALTER qualifier are:

- NAME=cachename − Name of the cache to be modified. The cache must already be defined in the database. This is a required parameter. This parameter accepts the wildcard characters asterisk (*) and percent sign (%).
- DROP − Remove the row cache definition from the database.
- SNAPSHOT_SLOT_COUNT=n − Specify the number of snapshot slots in the cache. A value of zero disables in−cache snapshots for the cache.
- SLOT_COUNT=n − Specify the number of slots in the cache.
- SLOT_SIZE=n − Specify the size (in bytes) of each slot in the cache.
- WINDOW_COUNT=n − Deprecated qualifier.
- WORKING_SET_COUNT=n − Specify the number of working set entries for the cache. Valid values are from 1 to 100. Note that this keyword is deprecated.
- BACKING_STORE_LOCATION=devdir − Specify the per−cache default backing store location.
- NOBACKING_STORE_LOCATION − Remove the per−cache default backing store location and revert back to the database default backing store file location.
- SHARED_MEMORY − Specify the shared memory type and parameters for the cache. Valid keywords are:
    - TYPE=PROCESS to specify traditional shared memory global section, which means that the database global section is located in process (P0) address space and may be paged from the processes working set as needed.
    - TYPE=RESIDENT to specify that the database global section is memory resident in process (P0) address space using OpenVMS Alpha shared page tables, which means that a system space global section is fully resident, or pinned, in memory.
    - RAD_HINT= "number" to indicate a request that memory for this shared memory should be allocated from the specified OpenVMS Alpha Resource Affinity Domain (RAD). This parameter specifies a hint to Oracle Rdb and OpenVMS about where

memory should be physically allocated. It is possible that if the memory is not available, it will be allocated from other RADs in the system. For systems that do not support RADs, no RAD_HINT specification is valid.
The RAD_HINT qualifier is only valid when the shared memory type is set to RESIDENT. Setting the shared memory type to SYSTEM or PROCESS explicitly disables any previously defined RAD hint.

---

Note

***OpenVMS support for RADs is available only on the AlphaServer GS series systems. For more information about using RADs, refer to the OpenVMS Alpha Partitioning and Galaxy Guide.***

---

◊ NORAD_HINT disables the RAD hint.
◊ TYPE=SYSTEM Deprecated keyword.

♦ SNAPSHOT_SLOT_COUNT=n − Specify the number of snapshot slots in the cache. A value of zero disables the snapshot portion for the specified cache. Otherwise, the total number of rows for the cache (the combination of "live" rows and snapshot rows) must be between 1 and 2,147,483,647.

The "/ALTER=(...)" qualifier may be specified multiple times on the command line. Each /ALTER qualifier specified operates on one unique cache if no wildcard character (% or *) is specified. Otherwise, each /ALTER operates on all matching cache names. For example, the following command alters two caches:

```
$ RMU /SET ROW_CACHE MF_PERSONNEL -
        /ALTER= (   NAME = RDB$SYS_CACHE,
                    SLOT_COUNT = 800, -
                    WINDOW_COUNT = 25 ) -
        /ALTER= (   NAME = RESUMES, -
                    SLOT_SIZE=500, -
                    WORKING_SET_COUNT = 15)
```

The following command alters caches named FOOD and FOOT (and any other cache with a 4 character name with the first three characters of "FOO" defined in the database):

```
$ RMU /SET ROW_CACHE MF_PERSONNEL -
        /ALTER= (   NAME = FOO%,
                    BACKING_STORE_LOCATION=DISK$RDC:[RDC])
```

The following example modifies the database MYDB to set the snapshot slot count for the cache "EMPL_IDX" to 250,000 slots and disables snapshots in cache for the "SALES" cache:

```
$ RMU /SET ROW_CACHE DGA0:[DB]MYDB.RDB -
  /ALTER=(NAME=EMPL_IDX, SNAPSHOT_SLOT_COUNT=250000) -
  /ALTER=(NAME=SALES, SNAPSHOT_SLOT_COUNT=0)
```

# 6.7 RMU Show Statistics Errors Fixed

## 6.7.1 RMU /SHOW STATISTICS Writes Invalid Configuration File

Bug 3108571

Starting in Oracle Rdb Release 7.0.6.3, the RMU /SHOW STATISICS utility could write an invalid line to a save configuration file. In particular, the "CHECKPOINT_TX" line would be omitted and the line containing "CHECKPOINT_BLOCK_COUNT" would be corrupted.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.7.2 RMU /SHOW STATISTICS /DEADLOCK_LOG Does Not Record First Deadlock Occurrence

Bug 3104519

Previously, it was possible for the first deadlock encountered by a process to not be correctly logged by the RMU /SHOW STATISTICS utility when using the /DEADLOCK_LOG qualifier.

The following sequence of events shows one possible way that this behaviour could be seen. Three sessions are needed; one to execute RMU /SHOW STATISTICS and two others to use interactive SQL.

```
$! In session 1
$ RMU /SHOW STATISTICS MF_PERSONNEL -
    /NOINTERACTIVE -
    /TIME=1 -
    /UNTIL="''F$CVTIME("+0:1","ABSOLUTE")'" -
    /DEADLOCK_LOG=DEAD.LOG

       session 2                            session 3

step1 $ SQL$                               $SQL$
      SQL> attach 'file mf_personnel';    SQL> attach 'file mf_personnel';

step2 SQL> select * from employees        SQL> select * from employees
            where  employee_id = '00165';       where employee_id = '00190';

step3 SQL> update employees
           set middle_initial = 'F'
           where employee_id = '00190';

step4                                      SQL> update employees
                                                set middle_initial = 'F'
                                                where employee_id = '00165';
```

After letting the RMU /SHOW STATISTICS session expire, examining the content of DEAD.LOG shows that it contains only the header information with no record of the deadlock.

This problem has been corrected in Oracle Rdb Release 7.1.2.

### 6.7.3 RMU /SHOW STATISTICS Row Cache Overview Integer Overflow

Bug 2644488

Statistics information maintained by Oracle Rdb are stored in longword integer counters. Some basic operations on these counters are treated as signed while others are treated as unsigned. In the "Row Cache Overview" display of the RMU /SHOW STATISTICS utility, it was possible for percentages to be incorrectly displayed when counters exceeded approximately 2,147,483,647.

This problem has been corrected for the "Row Cache Overview" display for Oracle Rdb Release 7.1.2. Some of the integer operations are performed using unsigned quadwords to avoid overflow resulting in negative values.

However, an integer longword can still represent approximately 4.29 billion values before it overflows back to zero. When a counter does overflow and "wraps" back to zero, comparisons and operations on this value may result in unexpected results when displayed by the RMU /SHOW STATISTICS utility.

Oracle anticipates correcting this behaviour in the future.

### 6.7.4 RMU /SHOW STATISTICS "Device Information" Screen Bugchecks in Playback Mode

Bug 2882453

RMU /SHOW STATISTICS bugchecks on selecting the "Device Information" option under the IO Statistics (by file) in playback mode. This screen was enhanced to display information about devices with AIJ files. The information necessary to do this is not saved in the binary file and hence cannot be played back.

This problem has been corrected in Oracle Rdb Release 7.1.2. A change has been made to display information of devices with AIJ files only in the normal mode.

### 6.7.5 RMU /SHOW STATISTICS Limiting Multi−Page Report

Previously, the RMU /SHOW STATISTICS utility would always write all pages for each display when writing to a report file. For some databases, for example those with a large number of row cache slots, this output would be quite lengthy.

As an assist to reducing the size of the output report file, the qualifier "MULTIPAGE_MAXIMUM=n" has been added to the RMU /SHOW STATISTICS command. When specified, the MULTIPAGE_MAXIMUM qualifier accepts a number representing the maximum number of pages to write for a multi−page display. For example, if /MULTIPAGE_MAXIMUM=5 is specified, only the first 5 pages of any multi−page display will be written to the output file. If MULTIPAGE_MAXIMUM is not specified, the current behaviour of "unlimited" is assumed and all pages for multi−page displays will be written to the output file.

## 6.7.6 RMU /SHOW STATISTICS Misleading Maximum Values After RESET or UNRESET

Bug 2790163

Previously, the determination of the maximum value for a statistics counter was incorrectly determined after a RESET or UNRESET operation. The amount of time that the statistic value represented was not being accurately calculated leading to an unexpectedly large or small peak value.

This problem has been corrected in Oracle Rdb Release 7.1.2. Statistics are now collected immediately prior to a RESET or UNRESET operation.

## 6.7.7 Enhancement to Transaction Duration Display When Written to Report

Bug 1982063

The "Write Report" function of the RMU /SHOW STATISTICS utility now includes all 3 variants of the "Transaction Duration" display: Read/Write, Read/Only and Combined.

## 6.7.8 RMU /SHOW STATISTICS Bugcheck in KUTDIS$UPDATE_RS

Bug 2912167

Previously, it was possible for the RMU /SHOW STATISTICS utility to bugcheck with an ACCVIO in the routine KUTDIS$UPDATE_RS when logical area statistics were being displayed. This problem was caused by an internal data structure being slightly undersized.

This problem has been corrected in Oracle Rdb Release 7.1.2. As a possible workaround, use the /NOLOGICAL_AREA qualifier.

## 6.7.9 RMU /SHOW STATISTICS 95th Percentile Transaction Duration Less Than 0.40 Seconds

Bug 2388494

At the completion of a transaction, Oracle Rdb captures the transaction duration and also updates a table of transaction durations. This table is used to display frequency of transactions by duration. In the past, the shortest transaction duration represented in this table was 0.40 seconds meaning that any transaction that completed in 0.40 seconds or less would be counted in the 0.40 seconds bucket. This effect can make it more difficult to optimize or tune database systems where most transactions are quite short. Also on modern, fast OLTP systems, many or most transactions may well be less than 0.40 seconds.

This behaviour has been changed in Oracle Rdb Release 7.1.2. The shortest transaction duration represented in the transaction duration table is now 0.01 seconds and the scaling of intervals within the table has been altered to provide finer detail of shorter transactions.

## 6.7.10 "RUJ File Writes" Statistic Not Accurate

Bug 2936595

Previously, the "RUJ file writes" database statistic as displayed by the RMU /SHOW STATISTICS utility on the "RUJ Statistics" screen was not being correctly maintained and would often display a value that was much too small.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.7.11 RMU /SHOW STATISTICS Custom (Yanked) Value Double of Value Reported in Original Screen

Bug 2907249

Previously, the "yanked" value as shown on the custom statistics display would be double the value reported on the original statistics display.

This problem has been corrected in Oracle Rdb Release 7.1.2.

# 6.8 Hot Standby Errors Fixed

## 6.8.1 Changes to RMU /REPLICATE AFTER START /BUFFERS Qualifier

Bug 2643337

In prior releases of Oracle Rdb, the ***/BUFFERS*** qualifier for the ***RMU/REPLICATE AFTER START*** command was ignored if the database was not configured to use global buffers. The database administrator had no control over the number of buffers used by the Log Recovery Server (LRS). The LRS would determine the number of buffers based on various database configuration parameters. While the value determined by the LRS was usually a reasonable number, occasionally it would not be optimum.

This behavior has been changed. The LRS will now use the value specified by the ***/BUFFERS*** qualifier, regardless of whether or not global buffers are being used.

In addition, the defaults for the ***/BUFFERS*** qualifier have been changed as follows:

***Table 6−1 / Buffer Values***

| | Local Buffers | Global Buffers |
|---|---|---|
| Default Value | 4096 | 4096 or Global Buffer USER LIMIT, whichever is smaller |
| Minimum Value | 2 | 2 |
| Maximum Value | 524,288 | 524,288 or Global Buffer USER LIMIT, whichever is smaller |

If global buffers are not being used on the standby database, and the currently configured buffer value is less than 4096, it is advised that after installing this release the ***RMU/REPLICATE AFTER CONFIGURE /MASTER /BUFFER*** command be used to explicitly set the number of buffers to an appropriate value.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 6.8.2 Standby Database Missing Updates if Master Not Manually Opened

Bug 2656517

When the following conditions are true, it is possible for database updates to not get propagated to a standby database:

- Hot Standby replication is enabled and started
- The database is set to OPEN IS AUTOMATIC instead of OPEN IS MANUAL as recommended in the documentation
- A user on a node that does not currently have the master database open attaches to the

database and updates it before the AIJ Log Server (ALS) process starts

In the above scenario, it is possible for journal entries to be written by a user process instead of the ALS process. When that occurs, the journal entries written by the user process are not transmitted to the standby database and thus are not reflected in the standby database.

The ALS process will be automatically started when a user attaches to a database, but before this release it was possible for a user process to update the database before the ALS process was ready to transmit those changes to the standby database.

To avoid this problem, always manually open databases that utilize the Hot Standby feature prior to making any updates to those databases.

This problem has been corrected in Oracle Rdb Release 7.1.2. The Oracle Rdb monitor process will now defer responding to user attach requests until the ALS process is ready.

## 6.8.3 Starting LRS on Master Database Caused Shutdown

Bug 1775983

If a database administrator attempted to start the Hot Standby feature and mistakenly specified the master database as the standby database then the master database was shutdown.

For example, note that in the following command the /MASTER qualifier is being used against the master database. This causes Hot Standby to attempt to configure the master database as a standby database.

```
$ RMU /REPLICATE AFTER_JOURNAL START [.MASTER]mf_personnel −
  /MASTER_ROOT=[.STANDBY]standby_personnel
```

When the above command was issued, the AIJ Log Roll−forward Server (LRS) failed with the following error:

```
RDMS−F−STBYDBINUSE, standby database cannot be
exclusively accessed for replication
```

Whenever the LRS failed, the database was automatically shutdown. A running application could be accidentally shutdown due to this type of command error.

This problem has been corrected in Oracle Rdb Release 7.1.2. Now, if the LRS fails with a STBYDBINUSE error, the database is not shutdown.

# 6.9 Oracle Trace Errors Fixed

## 6.9.1 Oracle Trace Did Not Collect for Oracle Rdb Release 7.1.1

Oracle TRACE did not collect data for the RDBVMS FACILITY for Oracle Rdb Release 7.1.1.0.0 and 7.1.1.0.1. This was because the version number specified by Rdb to the Oracle TRACE Service Routine EPC$INIT was "V7.1−10" but the version number inserted by the Rdb installation into the module "RDBVMSV7.1−10" in the TRACE VMS SYS$SHARE:EPC$FACILITY.TLB text library was "V7.1−100" or "V7.1−101". This module could then be extracted and used to set the RDBVMS facility definition and version in the Trace Administration database. The version string passed to the Oracle TRACE Service Routine EPC$INIT must match identically with the version specified in the facility definition in the Trace Administration database. Since this was not true for the RDBVMS facility, when an RDBVMS TRACE collection was scheduled, no RDBVMS data was collected.

For Oracle Rdb Release 7.1.2.0.0 and 7.1.2.0.1, this has been corrected and the version number passed to EPC$INIT for the RDBVMS facility will exactly match the RDBVMS facility version inserted by the Rdb installation procedure into SYS$SHARE:EPC$FACILITY.TLB which can then be used to define the RDBVMS facility in the TRACE Administration database. For both Rdb 7.1.2.0.0 and 7.1.2.0.1, this version is "V7.1−2".

The following example shows that for Oracle Rdb Release 7.1.1.0.0 and 7.1.1.0.1, RDBVMS facility data was not getting collected as signified by no arrow pointing to the RDBVMS facility when the Oracle TRACE "COLLECT SHOW REGISTER/NOCLUSTER" command is issued. Data is getting collected for the ATM_SAMPLE facility as indicated by the arrow.

```
$ COLLECT SHOW REGISTER/NOCLUSTER

Registrations actively collecting

  Node: CLNODE  Collection: SAMPLE            Selection: SAMPLE_SELECTION
                            _COLLECTION

    Process   Process Name     Facility  Version   Registration Id
    --------  ---------------  --------  --------  ----------------------------
    20638408  _RTA2:           RDBVMS    V7.1−101
 ->                            ATM       V1.2−0    ATM APPLICATION EXT
                               _SAMPLE
```

The only workaround for this problem would be to change the version numbers in the Oracle TRACE administration database tables from "V7.100" or "V7.101" to "V7.1−10".

# Chapter 7
# Enhancements

The Oracle Rdb RMU Reference Manual was updated with all the new and changed features for RMU up to and including Rdb Release 7.1.0.4 (updated in Sept. 2003). The Oracle Rdb SQL Reference Manual was updated with all the new and changed features for SQL up to and including Rdb Release 7.1.2 (updated in Feb. 2004). These enhancements to Oracle Rdb are no longer available in these release notes. Please refer to these updated documents for new features. The updated manuals can be accessed from the Rdb OTN Documentation Page or from MetaLink.

# 7.1 Enhancements Provided in Oracle Rdb Release 7.1.2.4

## 7.1.1 Query Governor Enhanced to Timeout Executing Requests

Bugs 809183 and 3240288

This release of Oracle Rdb introduces the ability to timeout long running database requests. A database request may include SELECT, UPDATE, DELETE and INSERT statements executed singly or as part of a compound statement, and includes any constraints or triggers associated with such actions.

Some database requests may take an excessive amount of time or system resources to execute. This feature provides the ability to have a request return an error if the request exceeds a defined limit for CPU or elapsed time.

Oracle Rdb provides the following interfaces to this timeout facility.

♦ SQL Interface
  New SQL syntax has been added to support this ability. An updated syntax diagram for SET QUERY is shown in Section 7.1.2, SET QUERY Statement. This new syntax is supported from both interactive and dynamic SQL interfaces. The following example establishes a 5 minute query timeout for the session.

  ```
  SET QUERY EXECUTION LIMIT ELAPSED TIME 5 MINUTES;
  ```

  For smaller intervals, the keyword SECONDS can be used, and is the default unit for this statement.

  ```
  SET QUERY EXECUTION LIMIT CPU TIME 30 SECONDS;
  ```
♦ Logical Name Interface
  Additionally, logical names may be used to define the timeout values. For example, to limit any database request that requires more than one second of CPU time or two seconds of elapsed time the following logical names can be defined.

  ```
  $ DEFINE RDMS$BIND_QG_EXEC_CPU_TIMEOUT 1
  $ DEFINE RDMS$BIND_QG_EXEC_ELAPSED_TIMEOUT 2
  ```

  When the timeout limit has been exceeded, Oracle Rdb will return an ***RDB−E−EXQUOTA*** error with a secondary error of ***RDMS−E−MAXTIMLIM***. For example:

  ```
  SQL> DELETE FROM EMPLOYEES;
  %RDB−E−EXQUOTA, Oracle Rdb runtime quota exceeded
  −RDMS−E−MAXTIMLIM, query governor maximum timeout has been reached
  ```
♦ RMU/SHOW STATISTICS Interface
  Requests may also be canceled using the ***RMU/SHOW STATISTICS*** utility. If a particular

database user has a database request that is taking an excessive amount of time, it may be canceled by a database administrator. The ***RMU/SHOW STATISTICS*** tools menu provides a mechanism to cancel a long running request. The tools menu is invoked by the "!" command. That menu contains the option "Terminate request". When that option is selected, ***RMU/SHOW STATISTICS*** prompts for the target user's process identifier (PID) and stream identifier. (The stream identifier is necessary if the user has more than one attach in that database.) The target process is then told to abort the currently running request. When the request is terminated, an ***RDB−E−EXQUOTA*** error is returned to the application; the secondary error code is ***RDMS−E−REQCANCELED***. For example:

```
SQL> DELETE FROM EMPLOYEES;
%RDB−E−EXQUOTA, Oracle Rdb runtime quota exceeded
−RDMS−E−REQCANCELED, request canceled
```

If an application receives a timeout error, it may continue to execute new database requests. The current transaction does not have to be terminated. New requests are still subject to the declared timeout limits.

Note that this timeout applies to the execution of a single database request. The timer is restarted for each new database request. A compound statement or stored procedure is treated as a single database request therefore, if a compound or stored procedure contains multiple statements or calls to nested stored routines, then all of the statements will be treated as a single database request.

Timeouts are not nested. If a compound statement includes an external function call and that external function issues a database request, the new database request issued by the external function becomes the current request being timed. When that external function returns, the compound statement that invoked the external function is no longer being timed. The exception is when the external function uses the BIND ON SERVER clause to force a separate server process to be used for the function invocation.

The timer is implemented using a low−overhead polling mechanism. That means that the timer is not completely accurate. For example, if a timeout is declared to be five seconds, a request may timeout in as little as five seconds but may not timeout for as long as ten seconds.

Timeouts are not applied to external routines which do not execute queries on the database. For instance, a compute bound function doing no I/O will not execute the polling required to terminate the request. Such external routines will not be terminated even if it exceeds the timeout values specified. However, once control returns to the calling request then the expired timer will be detected and the request terminated.

Timeouts only apply to basic query and data manipulation statements. Timeouts do not apply to database operations such as data definition statements (DDL). For example, a CREATE INDEX or an ALTER DATABASE operation will not timeout.

## 7.1.2 SET QUERY Statement

The SET QUERY statement is used to control query execution within a SQL session.

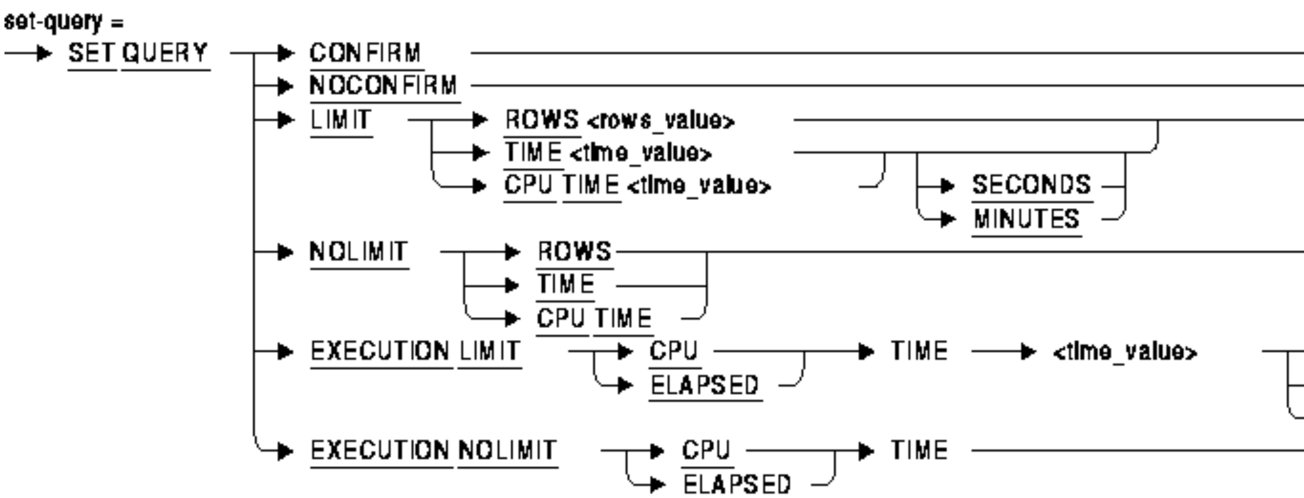
***Environment***

You can use the SET QUERY statement:

- In interactive SQL
- In dynamic SQL as a statement to be dynamically executed

Note that some options for the SET QUERY command may only be used in interactive SQL.

***Syntax:***




***Arguments:***

- CONFIRM
  Lets you preview the cost of a query, in terms of I/O, before any rows are actually returned. For example:

  ```
  SQL> SELECT * FROM EMPLOYEES;
  Estimate of query cost: 52 I/O s, rows to deliver: 100
  Do you wish to cancel this query (No)? YES
  %SQL−F−QUERYCAN, Query cancelled at user's request
  ```

  Some queries can result in Oracle Rdb performing a large number of I/O operations, retrieving a large number of rows, or both. The SET QUERY CONFIRM statement causes SQL to display estimated query costs. If the cost appears excessive, you can cancel the query by answering Yes; to continue, answer No.
  The SET QUERY CONFIRM statement is only available for interactive SQL.
- NOCONFIRM
  Disables the query confirm dialog that was previously enabled using SET QUERY CONFIRM. The SET QUERY NOCONFIRM statement is only available for interactive SQL.
- LIMIT
  Sets limits to restrict the output generated by a query.
  The mechanism used to set these limits is called the query governor. The following gives you three ways to set limits using the query governor:

- ◊ ROWS rows_value
  You can restrict output by limiting the number of rows a query can return. The optimizer counts each row returned by the query and stops execution when the row limit is reached.
  The default is an unlimited number of row fetches. Dynamic SQL defaults are inherited from the compilation qualifier for the module.
- ◊ TIME time_value [ SECONDS | MINUTES ]
  You can restrict the amount of time used to optimize a query for execution. If the query is not optimized and prepared for execution before the total elapsed time limit is reached, an error message is returned.
  The default is unlimited time for the query compilation. If you omit the SECONDS and MINUTES keyword then SECONDS is the default.

---

Note

***Specifying a query time limit can cause application failure in certain circumstances. For instance, an application that runs successfully during off−peak hours may fail when run during peak hours due to the load on the database.***

---

- ◊ CPU TIME time_value [ SECONDS | MINUTES ]
  You can restrict the amount of CPU time used to optimize a query for execution. If the query is not optimized and prepared for execution before the CPU time limit is reached, an error message is returned.
  The default is unlimited CPU time for the query compilation. If you omit the SECONDS and MINUTES keyword then SECONDS is the default. Dynamic SQL options are inherited from the compilation qualifier for the module.

Use a positive integer for the number of rows and the number of seconds; negative integers are invalid and zero means no limits. If an established limit is exceeded, the query is canceled and an error message is displayed. When you set both a time limit and the row limit, whichever value is reached first stops the output.
Application developers can use this feature to prevent users from overloading the system. The database administrator can manage system performance and reduce unnecessary resource usage by setting option limits.

- ♦ NOLIMIT
  This option removes a limit imposed by the SET QUERY LIMIT command.
  Use one of the following options.
  - ◊ TIME
  - ◊ CPU TIME
  - ◊ ROWS

  NOLIMIT is equivalent to assigning a limit of zero to any of the options using SET QUERY LIMIT.
- ♦ EXECUTION LIMIT
  This option imposes elapsed and CPU time limits on executing queries. This command affects all subsequent queries executed within the Rdb server process. You must be attached to a database to execute this statement. This statement affects all attaches for the current process, not just the current connection.
  - ◊ ELAPSED TIME time_value [ SECONDS | MINUTES ]

◊ CPU TIME time_value [ SECONDS | MINUTES ]

You can restrict the amount of elapsed time or CPU time used to execute a query. If the query is not complete before the elapsed or CPU time limit is reached, an error message is returned. The default is unlimited time for the query execution. If you omit the SECONDS and MINUTES keyword then SECONDS is the default. Dynamic SQL options are inherited from the compilation qualifier for the module.

---

Note

***Specifying a query time limit can cause application failure in certain circumstances. For instance, an application that runs successfully during off−peak hours may fail when run during peak hours due to the load on the database.***

---

Use a positive integer for the number of seconds or minutes; negative integers are invalid and zero means no limits. If an established limit is exceeded, the query is canceled and an error message is displayed. When you set a CPU time limit, elapsed time limit and a row limit (using SET QUERY LIMIT), whichever value is reached first stops the query.
Database administrators and application developers can use this feature to prevent users from overloading the system by executing long running, and probably unproductive queries. The database administrator can manage system performance and reduce unnecessary resource usage by setting option limits.

♦ EXECUTION NOLIMIT
This option removes a limit imposed by the SET QUERY EXECUTION LIMIT command. Use one of the following options.
  ◊ ELAPSED TIME
  ◊ CPU TIME

EXECUTION NOLIMIT is equivalent to assigning a limit of zero to any of the options using SET QUERY EXECUTION LIMIT.

♦ time_value
This argument represents the number of seconds or minutes specified for the SET QUERY statement. It can be a numeric literal, a parameter name (for interactive SQL), or a parameter−marker (for dynamic SQL).

♦ rows_value
This argument represents the number of rows specified for the SET QUERY statement. It can be a numeric literal, a parameter name (for interactive SQL), or a parameter−marker (for dynamic SQL).

***Examples***

This example shows the syntax for establishing a row limit within an interactive SQL session.

```
SQL> set query limit rows 10000;
SQL> show query limit;
QUERY LIMIT TIME is OFF
QUERY LIMIT ROWS limit is 10000 rows
QUERY LIMIT CPU TIME is OFF
SQL> set query nolimit rows ;
SQL> show query limit;
QUERY LIMIT TIME is OFF
```

```
QUERY LIMIT ROWS is OFF
QUERY LIMIT CPU TIME is OFF
```

This example uses SET QUERY to establish a 2 second elapsed time limit for a query, and shows the error message that is displayed.

```
SQL> set query execution limit elapsed time 2 seconds;
SQL> delete from EMPLOYEES;
%RDB−E−EXQUOTA, Oracle Rdb runtime quota exceeded
−RDMS−E−MAXTIMLIM, query governor maximum timeout has been reached
SQL> set query execution nolimit elapsed time;
```

## 7.1.3 RMU /POPULATE_CACHE Command

In previous versions of Oracle Rdb, it was difficult to cause all nodes of an index or all rows of a table to be loaded into a memory cache (either global buffers or row caches) upon demand. Most applications naturally cause frequently accessed data to be cached upon the first read. However, for some classes of applications, it is desirable to cause all existing index nodes or table rows to be explicitly fetched from the database into cache(s).

The RMU /POPULATE_CACHE command allows one or more tables and indexes to be read from the database and stored in caches (if they exist).

Sorted indexes are read "top down", one index level at a time. Hashed indexes are read by sequentially scanning the storage area(s) containing the hashed indexes and fetching all nodes and the system record from each database page. Data table rows are read by sequentially scanning the storage area(s) containing the table and fetching all rows of the relation.

Table 7−1 describes the command qualifiers for the RMU /POPULATE_CACHE command.

***Table 7−1 RMU / POPULATE_CACHE Command Qualifiers***

| Qualifier | Description |
|---|---|
| /TABLE=table−list | Specifies names of one or more tables to fetch. All rows are fetched from each table. If you list multiple tables, separate the table names with a comma and enclose the list within parentheses. Wildcard characters "*" and "%" are allowed. |
| /INDEX=index−list | Specifies names of one or more indexes to fetch. All nodes are fetched from each index. If you list multiple indexes, separate the index names with a comma and enclose the list within parentheses. Wildcard characters "*" and "%" are allowed. |
| /LOG | Specifies whether the processing of the command is reported to SYS$OUTPUT. Specify the Log qualifier to request that information about the operation be displayed. If you specify neither /NOLOG nor /LOG, the default is the current setting of the DCL verify switch. (The DCL SET VERIFY command controls |

| | |
|---|---|
| | the DCL verify switch.) |
| /TRANSACTION_TYPE=transaction_mode | Allows you to specify the transaction mode, isolation level, and wait behavior for transactions. Use one of the following keywords to control the transaction mode:<br>♦ AUTOMATIC − When Transaction_Type=Automatic is specified, the transaction type depends on the current database settings for snapshots (enabled, deferred, or disabled), transaction modes available to this user, and the standby status of the database. Automatic mode is the default.<br>♦ READ_ONLY − Starts read−only transactions.<br>♦ WRITE − Starts read−write transactions. |

The following examples demonstrate several RMU /POPULATE_CACHE commands.

```
$ RMU /POPULATE_CACHE /INDEX=* /LOG MF_PERSONNEL
%RMU−I−NOCCH, No cache present for "DEPARTMENTS_INDEX"
%RMU−I−NOCCH, No cache present for "EMPLOYEES_HASH"
%RMU−I−NOCCH, No cache present for "JOB_HISTORY_HASH"
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
 ELAPSED:    0 00:00:00.01  CPU: 0:00:00.02  BUFIO: 20  DIRIO: 63  FAULTS: 265
Index "COLL_COLLEGE_CODE" : 1 node fetched
Index "DEG_COLLEGE_CODE" : 14 nodes fetched
Index "DEG_EMP_ID" : 71 nodes fetched
    .
    .
    .
Index "RDB$VIEW_REL_NAME_NDX" : 4 nodes fetched
Index "RDB$VIEW_VIEW_NAME_NDX" : 4 nodes fetched
Index "SH_EMPLOYEE_ID" : 99 nodes fetched
Total : 892 fetched
$
$ RMU /POPULATE_CACHE /INDEX=*REL* /LOG MF_PERSONNEL
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
 ELAPSED:    0 00:00:00.00  CPU: 0:00:00.01  BUFIO: 9  DIRIO: 7  FAULTS: 180
Index "RDB$CR_REL_NAME_NDX" : 9 nodes fetched
Index "RDB$NDX_REL_NAME_NDX" : 23 nodes fetched
    .
    .
    .
Index "RDB$VER_REL_ID_VER_NDX" : 39 nodes fetched
Index "RDB$VIEW_REL_NAME_NDX" : 4 nodes fetched
Total : 122 fetched
$
$ RMU /POPULATE_CACHE /INDEX=(EMP_LAST_NAME,EMP_EMPLOYEE_ID) −
    /TABLE=(C1,C3) /NOLOG MF_PERSONNEL
$
$ RMU /POPULATE_CACHE /TABLE=C% /LOG MF_PERSONNEL
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
 ELAPSED:    0 00:00:00.04  CPU: 0:00:00.03  BUFIO: 9  DIRIO: 239  FAULTS: 328
Table "C1" : 30 rows fetched
Table "C2" : 5 rows fetched
Table "C3" : 5 rows fetched
Table "C4" : 5 rows fetched
```

```
Table "C5" : 0 rows fetched
Total : 45 fetched
```

## 7.1.4 RMU Support for /DENSITY = SDLT320

Oracle Rdb RMU commands that support the /DENSITY qualifier (ie, RMU/BACKUP, RMU/BACKUP/AFTER_JOURNAL and RMU/OPTIMIZE_AIJ) now support the keyword "SDLT320" for use with SuperDLT320 tape drives.

## 7.1.5 RMU Delete Optimizer_Statistics Supports /TABLE=*

Bug 3522877

This release of Oracle Rdb allows RMU/DELETE OPTIMIZER_STATISTICS to accept (*) as a wildcard for all tables registered in the RDB$WORKLOAD table. This allows the RDB$WORKLOAD table to be purged.

The following example shows that all RDB$WORKLOAD rows are removed by this command.

```
$ rmu/delete optimizer/log sql$database/table=*
%RMU-I-COLTXT_07, All workload column groups deleted for EMPLOYEES
%RMU-I-COLTXT_07, All workload column groups deleted for JOB_HISTORY
%RMU-I-COLTXT_07, All workload column groups deleted for RDB$FIELDS
%RMU-I-COLTXT_07, All workload column groups deleted for RDB$OBJECT_SYNONYMS
%RMU-I-COLTXT_07, All workload column groups deleted for RDB$RELATIONS
%RMU-I-COLTXT_07, All workload column groups deleted for RDB$RELATION_FIELDS
```

# 7.2 Enhancements Provided in Oracle Rdb Release 7.1.2.3

## 7.2.1 Rdb Optional Site−Specific Startup Procedure

The Oracle Rdb startup procedure RMONSTART(xx).COM now supports an optional site−specific startup procedure to be executed after the Rdb Monitor (RDMMON) process has been started. If the file SYS$STARTUP:RDB$SYSTARTUP(xx).COM (where xx indicates the version number for multi−version Rdb kits) is found, it is executed as a DCL command procedure by the RMONSTART(xx).COM procedure.

SYS$STARTUP:RDB$SYSTARTUP(xx).COM is intended to contain site−specific tasks to be executed after the Rdb monitor procedure has completed. Such tasks might include opening databases or starting layered products that depend on the Rdb monitor process having been started.

If a site wishes to use this capability, the RDB$SYSTARTUP(xx).COM procedure must be created in SYS$STARTUP (either in the common SYS$COMMON:[SYS$STARTUP] directory or a node−specific SYS$SPECIFIC:[SYS$STARTUP] directory. The Rdb installation procedure does not provide or replace this file.

# 7.3 Enhancements Provided in Oracle Rdb Release 7.1.2

## 7.3.1 RMONSTOP71.COM Parameter for RMU /MONITOR STOP Command

Bug 3098311

The Oracle Rdb RMONSTOP71.COM procedure used to shutdown Oracle Rdb now accepts an optional parameter that can be used to control how the Rdb monitor is stopped. The parameter P1 can be passed as the string "/ABORT=DELPRC" or "/ABORT=FORCEX". This parameter is included in the RMU /MONITOR STOP command and can be used to cause the monitor process to terminate attached database users.

## 7.3.2 RMU/UNLOAD/AFTER_JOURNAL Output Flush

Bug 2832044

When using both the "OUTPUT" and "STATISTICS_INTERVAL" qualifiers with the RMU/UNLOAD/AFTER_JOURNAL command, the output stream used for the log, trace and statistics information is now flushed to disk (via the RMS $FLUSH service) at each statistics interval. This enhancement helps make sure that an output file of trace and log information is written to disk periodically.

## 7.3.3 RMU /SHOW STATISTICS Enhanced to Show Large Memory Setting

Bug 2903442

Previously, the RMU /SHOW STATISTICS "Buffer Information" screen did not display the setting of the global buffers "LARGE MEMORY IS ENABLED" setting.

This problem has been corrected in Oracle Rdb Release 7.1.2. The RMU /SHOW STATISTICS "Buffer Information" screen now indicates the database global buffers "LARGE MEMORY IS ENABLED" setting.

## 7.3.4 Statistics Collection Performance Improvement for AlphaServer GS Systems

NUMA (non−uniform memory access) is an attribute of a system in which access time to any given physical memory location is not the same for all CPUs. Given this architecture, consistently good location is important (but not necessarily 100 percent of the time) for highest performance. In the AlphaServer GS series, CPUs access memory in their own quad building block (QBB) faster than they access memory in another QBB. The OpenVMS operating system treats the hardware as a set of resource affinity domains (RADs). A RAD is a set of hardware components (CPUs, memory, and I/O) with common access characteristics. On AlphaServer GS80/160/320 systems, a RAD corresponds to a QBB.

Previously, a single copy of Oracle Rdb statistical information was maintained in a per−database memory structure (located in the database shared memory section). There was one copy of the statistical information for each database for all users on one OpenVMS system. Under heavy loads, the NUMA effect while maintaining statistics information could reduce the absolute performance of an application using Oracle Rdb due, in part, to increased memory access latency and CPU cache flushes.

The impact of this effect has been reduced. On AlphaServer GS series systems with more than one QBB configured with physical memory, the Oracle Rdb monitor process creates one global section per RAD that contains physical memory for the statistical information memory structure. These per−RAD global sections are created as "resident" and are requested to be allocated in physical memory by RAD. As each user attaches to the database, the user's OpenVMS defined "home" RAD is used to determine which global section to use for statistics collection for the user. The statistics global section is always mapped into the process' P0 virtual address space (ie, this global section is not controlled by SHARED MEMORY IS SYSTEM or LARGE MEMORY IS ENABLED).

---

Note

> ***The global section creation requested in physical memory of a specific RAD is simply a "hint" to OpenVMS. Memory may be obtained from other RADs if no free memory is available at the time of allocation.***

---

The RMU /SHOW STATISTICS utility maps all statistics global sections for a database. At each statistics collection interval, the statistical counters from each of the RAD−specific global sections are accumulated before display. Adding several copies of the statistics values together potentially increases the CPU consumption of the RMU /SHOW STATISTICS utility at each sample interval. However, the run−time performance gain by all database users should out−weigh the additional CPU cost of the RMU /SHOW STATISTICS utility. Using a less−frequent update interval in the RMU /SHOW STATISTICS utility will result in less CPU consumption as well.

The virtual memory consumed by processes attached to databases, the Oracle Rdb monitor (RDMMON) and the RMU /SHOW STATISTICS utility will increase on those systems with more than one QBB containing physical memory. This is due to the mapping of multiple statistics shared memory global sections. However, because these sections are physically resident in memory, additional working set quota should not be required. The amount of additional virtual address space consumed is proportional to the number RADs configured in the system, the number of storage areas, the number of logical areas and the number of row caches configured in each database.

---

Note

> ***OpenVMS support for RADs is available only on the AlphaServer GS series systems. For more information about using RADs, refer to the OpenVMS Alpha Partitioning and Galaxy Guide.***

---

## 7.3.5 RMU RECOVER Accepts Wildcard After−image Journal File Specifications and ORDER_AIJ_FILES Qualifier

Bug 3032437

Starting with Oracle Rdb Release 7.1.2, the RMU /RECOVER command accepts wildcard after−image journal file specifications. File specifications containing the OpenVMS wildcard characters "%" and "*" are now parsed and processed.

By default, after−image journal files are processed in the order that they are presented to the RMU RECOVER command (either explicitly or as returned from OpenVMS when using wildcards). The new ORDER_AIJ_FILES qualifier specifies that the input after−image journal files are to be processed in ascending order by sequence number. This can be of benefit when you use wildcard (* or %) processing of a number of input files. The .AIJ files are each opened, the first block is read (to determine the sequence number), and the files are closed prior to the sequence number sorting operation.

## 7.3.6 RMU/SHOW STATISTICS Page Dump Content and Format Enhancements

Bug 2899761

Starting with Oracle Rdb Release 7.1.2, the RMU /SHOW STATISTICS utility displays the database page content, including the content of rows on the page in the "PageInfo" display, if the user has the OpenVMS BYPASS privilege enabled. The display of the page content is now also consistant with the output format of the RMU /DUMP command.

## 7.3.7 Enhancement to Prestarted Transaction Timeout

Bug 2439694

Oracle Rdb Release 7.1 introduced the ability to timeout "prestarted" transactions. That functionality has been enhanced to also force a process to obtain a new transaction sequence number (TSN) if the same TSN has been reused throughout the duration of the prestarted transaction timeout interval.

When a READ WRITE transaction does not make any modifications to the database, Oracle Rdb will reuse the TSN for the next transaction. If a user rarely or never makes any database modifications then the TSN for the user will become old. This can cause snapshot files to grow excessively. This enhancement provides the ability for processes that constantly reuse TSNs to periodically obtain a new TSN, thus preventing excessive snapshot growth.

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 7.3.8 RDM$BIND_SNAP_QUIET_POINT Logical No Longer Used

Bug 2656534

If the logical RDM$BIND_SNAP_QUIET_POINT was defined to be "0" on a system that was used for the standby database in a Hot Standby configuration, it was not possible to start database replication. Attempts to start replication would fail with:

```
RDMS−F−HOTSNAPQUIET, quiet points must be enabled
for snapshot transactions during hot standby replication
```

However, defining the logical to "1" can cause processes with long running READ ONLY transactions to prevent database backups from proceeding.

This logical was introduced in Oracle Rdb Release 6.0 to allow database administrators to override the 6.0 requirement that READ ONLY transactions hold the quiet−point lock. Defining the logical to "1" (the default) would provide better performance when the Fast Commit feature was enabled and processes frequently switched between READ ONLY and READ WRITE transactions. Since that time, improvements have been made to the quiet−point lock algorithms that make this logical no longer necessary. Since releases 7.0.6.3 and 7.1.0.1, READ ONLY transactions would continue to hold the quiet−point lock until a backup process requested the lock. When the lock was requested, READ ONLY processes would release the quiet point lock as soon as the currently executing database request finished, if the RDM$BIND_SNAP_QUIET_POINT logical was defined as "0". This made it no longer necessary to have the logical defined as "1".

Since the quiet−point lock behavior now behaves optimally, even with the Fast Commit feature enabled, the RDM$BIND_SNAP_QUIET_POINT logical is no longer needed and thus has been removed. Oracle Rdb will now behave as if the logical is always defined to be "0".

This problem has been corrected in Oracle Rdb Release 7.1.2.

## 7.3.9 RMU Load Now Supports SELECTIVITY Option for OPTIMIZE Qualifier

The /OPTIMIZE qualifier for RMU Load now supports a new SELECTIVITY qualifier so that the Rdb query optimizer can be influenced to use different selectivity values.

The SELECTIVITY option accepts the following keywords:

- AGGRESSIVE − assumes a smaller number of rows will be selected (compared to the default Rdb selectivity)
- SAMPLED − uses literals in the query to perform preliminary estimation on indices
- DEFAULT − uses default selectivity rules

The following example shows how to use this new option.

```
$ RMU/UNLOAD/OPTIMIZE=(TOTAL_TIME,SELECTIVITY=SAMPLE) -
_$      SALES_DB CUSTOMER_TOP10 TOP10.UNL
```

This option is most useful when the RMU Unload command references a view definition with a complex predicate.

## 7.3.10 New Options Supported for LOGMINER SUPPORT Clause

The LOGMINER SUPPORT clause for CREATE DATABASE, IMPORT DATABASE, and ALTER DATABASE now allows the continuous mode for LogMiner to be enabled and disabled.

- LOGMINER SUPPORT IS ENABLED (CONTINUOUS)
  Enables continuous LogMiner.
- LOGMINER SUPPORT IS ENABLED (NOT CONTINUOUS)
  Disables continuous LogMiner, but leaves LogMiner enabled.
- LOGMINER SUPPORT IS DISABLED
  Disables LogMiner, including disabling continuous LogMiner.

Please refer to the Oracle Rdb RMU Reference Manual for more information about the Rdb LogMiner feature.

## 7.3.11 Changes to the IMPORT Command

In prior releases, the IMPORT command would display messages about the database. This is no longer true starting with Oracle Rdb Release 7.1.2. An example follows:

```
SQL> export data file mf_personnel into x;
SQL> drop data file mf_personnel;
SQL> import data from x file mf_personnel;
```

However, there is still the ability to generate these informational messages. A new clause, BANNER, has been added to the IMPORT command. Now, to enable informational messages to be displayed (hence the old behavior), simply specify BANNER on the IMPORT command line. Here is an example:

```
SQL> import data from x file mf_personnel BANNER;
Exported by Oracle Rdb X7.1−00 Import/Export utility
A component of Oracle Rdb SQL X7.1−00
Previous name was mf_personnel
It was logically exported on 29−MAY−2003 12:32
Multischema mode is DISABLED
Database NUMBER OF USERS is 50
Database NUMBER OF CLUSTER NODES is 16
   .
   .
   .
IMPORTing STORAGE AREA: EMPIDS_OVER
IMPORTing STORAGE AREA: EMP_INFO
IMPORTing STORAGE AREA: JOBS
IMPORTing STORAGE AREA: MF_PERS_SEGSTR
IMPORTing STORAGE AREA: SALARY_HISTORY
IMPORTing table CANDIDATES
IMPORTing table COLLEGES
IMPORTing table DEGREES
IMPORTing table DEPARTMENTS
IMPORTing table EMPLOYEES
IMPORTing table JOBS
IMPORTing table JOB_HISTORY
IMPORTing table RESUMES
IMPORTing table SALARY_HISTORY
```

```
IMPORTing table WORK_STATUS
IMPORTing view CURRENT_SALARY
IMPORTing view CURRENT_JOB
IMPORTing view CURRENT_INFO
```

It is also valid to specify NO BANNER on the IMPORT command line. Specifying NO BANNER is equivalent to specifying the IMPORT command with no clauses; that is informational messages about the database will not be displayed. In the following example, note that no informational messages are displayed:

```
SQL> import data from x file mf_personnel NO BANNER;
```

Warning and error messages will continued to be displayed as in prior releases.

## 7.3.12 New Warning Generated When Row Size Exceeds Row Cache Length

Bug 2909840

When a table's row length exceeds the size allocated for the associated logical area row cache, Rdb generates a warning to alert the database administrator of a possible problem with the row cache.

Rows can change size when ALTER TABLE adds a new column or alters an existing column's data type. An existing column can also be implicitly altered by ALTER DOMAIN, as the domain change is propagated to all referencing tables.

The following example shows the new warning message.

```
SQL> alter table NEW_EMPLOYEES alter column MIDDLE_INITIAL varchar(20);
%RDB−W−META_WARN, metadata successfully updated with the reported warning
−RDMS−I−RCHLENEXC, new row length exceeds size of row cache − check cache
attributes
```

Note that this ALTER TABLE operation succeeded. The warning is generated because rows that are too long to fit in the cache will not benefit from fast in−memory access. However, it is also quite possible that row compression will continue to result in compressed rows that still fit within the row cache. The database administrator will need to evaluate the change in light of the database design.

## 7.3.13 Oracle Media Management V2.0 API for Oracle Rdb RMU

The Oracle Media Management V2.0 API is supported by Oracle Rdb for RMU commands which accept the /LIBRARIAN qualifier. This support permits backing up to and restoring from, data archiving software applications supporting this interface. Examples of such applications include:

- Archive Backup System for OpenVMS from Hewlett−Packard Corporation on the world−wide−web at *http://h71000.www7.hp.com/openvms/storage/abspage.html*
- LEGATO NetWorker(R) from LEGATO Systems, Inc. on the world−wide−web at *http://www.legato.com/*
- Archive Backup Client (ABC) for OpenVMS from STORServer Inc. on the world−wide−web

at *http://www.storserver.com/*

More information on these products is available from the vendors.

### 7.3.13.1 Commands Accepting /LIBRARIAN

The Oracle Rdb Release 7.1.2 RMU commands which accept the /LIBRARIAN qualifier for storing data to this interface are:

- RMU /BACKUP
- RMU /BACKUP /AFTER_JOURNAL
- RMU /OPTIMIZE /AFTER_JOURNAL

The Oracle Rdb Release 7.1.2 RMU commands which accept the /LIBRARIAN qualifier for retrieving data from this interface are:

- RMU /RESTORE
- RMU /RESTORE /ONLY_ROOT
- RMU /DUMP /AFTER_JOURNAL
- RMU /DUMP /BACKUP_FILE
- RMU /RECOVER

RMU only supports the retrieval using the /LIBRARIAN qualifier for data that has been previously stored by RMU using the /LIBRARIAN qualifier.

In addition to the /LIBRARIAN qualifier used with existing RMU commands, there is a new RMU /LIBRARIAN /LIST command to list data streams stored in a LIBRARIAN implementation that have been created by RMU from a backup filename and a new RMU /LIBRARIAN /REMOVE command to delete data streams stored in a LIBRARIAN implementation that have been created by RMU from a backup filename.

---

Oracle Media Manager V2.0 Interface

***Only applications that conform to the Oracle Media Manager V2.0 specification can be called using the /LIBRARIAN qualifier or the new RMU /LIBRARIAN commands.***

---

RMU commands used with the /LIBRARIAN qualifier may not specify a list of tape or disk devices. It accepts a backup file ( "rbf file") name. Any disk or device specification and version number specified with the backup file name is ignored for the backup file name specified to the archive. For example, "device:[directory]FILENAME.RBF;1" is specified as "FILENAME.RBF " when the backup file data is stored in or retrieved from the archive.

### 7.3.13.2 Opaque Archive Application

The archive application is effectively an opaque "black box" in regards to RMU and the backup file name is the identifier of the stream of data stored in the archive. The utilities and command procedures specific to the particular LIBRARIAN application must be used to associate devices with the stream of data sent to or retrieved from the archive by RMU. Since the LIBRARIAN application is an opaque utility to RMU that stores and manages data, device specific qualifiers such as /REWIND, /DENSITY or /LABEL cannot be used with this interface.

### 7.3.13.3 RMU Backup Streams

Each writer thread for a backup operation or reader thread for a restore operation manages its own stream of data. Therefore, each thread uses a unique backup file name generated from the backup file name specified on the command line. A number is incremented and added to the end of the backup file extension (the extension defaults to ".RBF") specified to the archive (except for the first) representing a unique data stream. This number is the equivalent of the volume number associated with non−LIBRARIAN RMU backups and restores.

For example, if

```
$RMU /BACKUP /LIBRARIAN=(WRITER_THREADS=3) /LOG DB FILENAM.RBF
```

is specified for a backup command, the backup file data stream names

```
FILENAME.RBF
FILENAME.RBF02
FILENAME.RBF03
```

are specified to the archive to identify the three streams of data stored in the archive by the three writer threads which together represent the stored database. Since each data stream must contain at least one database storage area and a single storage area must be completely contained in one data stream, if the number of writer threads specified is greater than the number of storage areas, it will be set equal to the number of storage areas.

When

```
$RMU /RESTORE /LIBRARIAN=(READER_THREADS=3) /LOG FILENAM.RBF
```

is specified to restore the database, these same three data stream backup file names, one name specified by each of the three reader threads, will be generated by RMU and sent to the archive application to retrieve all the data associated with the database. If the number of reader threads is less than the number of backup writer threads, one or more restore reader threads will restore more than one data stream. If the number of reader threads specified is greater than the number of backup writer threads, the number of reader threads will be set equal to the number of backup writer threads so that all restored data is retrieved.

Therefore, the same number of reader threads in the example was specified on the restore as writer threads on the backup to generate all the stream names which represent the database. The user does not have to specify the same number of reader threads on the restore as writer threads specified on the backup. If a smaller number of reader threads on the restore is specified than the number of writer threads specified in the backup of the database, the data streams to be retrieved will be divided among the specified reader threads using an algorithm which assigns the data streams so that each thread will have an approximately equal amount of work to do. If a greater number of reader threads is specified on the restore than was specified on the backup, the number of reader threads will be automatically changed to equal the number of writer threads used in the backup.

The /VOLUMES qualifier cannot be used on the RMU/RESTORE command if the /LIBRARIAN qualifier is used. RMU automatically determines the number of data streams stored in the LIBRARIAN implementation based on the backup file name specified for the restore command and sets the volume number to the actual number of stored data streams. This helps to ensure that all data streams which represent the database are retrieved.

The default value for both WRITER_THREADS and READER_THREADS is "1". The WRITER_THREADS parameter can only be specified with the /LIBRARIAN qualifier for the RMU/BACKUP database command. The READER_THREADS parameter can only be specified with the /LIBRARIAN qualifier for the RMU /RESTORE database and the RMU /DUMP /BACKUP commands. All other RMU commands that accept the /LIBRARIAN qualifier only use one writer thread or one reader thread representing one archive data stream.

### 7.3.13.4 Parallel Backup Operations

The /LIBRARIAN qualifier can be used for parallel backup operations where backup threads can execute in multiple processes distributed among one or more nodes in a cluster. The database backup command can be invoked as a parallel command which uses multiple processes but the other RMU commands which accept the /LIBRARIAN qualifier do not support parallel processes but execute in one process.

The following lines in the backup PLAN file used to specify the parameters for parallel backup operations relate directly to the LIBRARIAN feature.

```
    Backup File = MF_PERSONNEL.RBF
    Style = Librarian
    Librarian_trace_level = #
    Librarian_trace_file = FILE.TRACE
    Librarian_logical_names = (−
             logical_name_1=equivalence_value_1, −
             logical_name_2=equivalence_value_2)
    Writer_threads = #
```

The backup file name must be the same file name specified for the restore and the style must be set to "Librarian" indicating a backup to the LIBRARIAN. The "Librarian_logical_names" entry is a list of logical names and their equivalence values. This is an optional parameter provided so that any logical names used by a particular LIBRARIAN application can be defined as process logical names before the backup or restore operation begins. For example, some LIBRARIAN applications provide support for logical names for specifying catalogs or debugging. "Librarian_trace_level = #" and "Librarian_trace_file = FILE.TRACE" are optional parameters specified with the /LIBRARIAN qualifier and passed to the LIBRARIAN application to be used for diagnostic purposes. The "Writer_threads = #" specifies the number of writer threads which will be used by each worker executor process. If this number exceeds the number of database storage areas assigned to a worker process, it will be set equal to the number of storage areas specified for that worker process.

If the backup is a parallel operation, a PLAN file is created and executed as part of the existing RMU /BACKUP /PARALLEL and RMU /BACKUP /PLAN command syntax. The following is an example of a parallel backup and non−parallel restore (the restore is always non−parallel and executes in a single process) using the /LIBRARIAN qualifier.

```
$RMU /BACKUP /PARALLEL=EXECUTOR=3 /LIBRARIAN=WRITER_THREADS=3−
 /LIST_PLAN=FILENAME.PLAN /NOEXECUTE /LOG DATABASE FILENAM.RBF
$RMU /BACKUP /PLAN FILENAME.PLAN
$RMU/ RESTORE /LIBRARIAN=(READER_THREADS=9) /LOG FILENAME.RBF
```

In this example, the first backup command creates the PLAN file for a parallel backup but does not execute it. The second backup command executes the parallel backup using the PLAN file. Note that

3 worker processes will be used and each process will use the 3 writer threads specified with the /LIBRARIAN qualifier. Each writer thread in each process will write one stream of backup data to the LIBRARIAN. Therefore 9 streams will be written to the LIBRARIAN archive. The streams will be given the names:

```
FILENAME.RBF
FILENAME.RBF02
FILENAME.RBF03
FILENAME.RBF04
FILENAME.RBF05
FILENAME.RBF06
FILENAME.RBF07
FILENAME.RBF08
FILENAME.RBF09
```

To retrieve the same 9 data streams which represent the backed up Rdb database on the non−parallel restore, a READER_THREADS=9 parameter can be specified with the /LIBRARIAN qualifier to use 9 threads to execute the restore, or if a READER_THREADS value between 1 and 8 is specified (1 is the default), RMU will determine the number of data streams actually stored by querying the LIBRARIAN implementation and distribute the data streams among the requested reader threads. If a READER_THREADS value is specified that is greater than "9", RMU will set it to "9" so that the restore does not attempt to retrieve data streams which do not exist.

## 7.3.13.5 Data Stream Naming Considerations

Since data stream names representing the database are generated based on the backup file name specified for the RMU backup command used with the /LIBRARIAN qualifier, the user must either use a different backup file name to store the next backup of the database to the LIBRARIAN implementation or first delete the existing data streams generated from the backup file name before the SAME backup file name can be reused for the next backup.

To delete the existing data streams stored in the LIBRARIAN implementation, a LIBRARIAN management utility can be used or the RMU /LIBRARIAN /REMOVE command described below can be used with just the backup file name to delete all the data streams generated based on that name. The user can incorporate the date or some other unique identifier in the backup file name when he does each backup to make it unique if he wants to avoid deleting a previous backup to the LIBRARIAN which used the same backup file name. Many LIBRARIAN implementations allow the user to specify an automatic deletion date for each data stream stored in their archives.

## 7.3.13.6 /LIBRARIAN Parameters

The /LIBRARIAN qualifier accepts the following parameters.

- WRITER_THREADS=#
  Use # writer threads to write # backup data streams to the LIBRARIAN. The database storage areas will be partitioned among the database streams. The streams will be named BACKUP_FILENAME.EXT, BACKUP_FILENAME.EXT02, BACKUP_FILENAME.EXT03, up to BACKUP_FILENAME.EXT99. BACKUP_FILENAME.EXT is the backup file name specified in the RMU command excluding any specified device, directory or version number. The default extension name is ".RBF". The "WRITER_THREADS" parameter can only be specified for parallel and non−parallel database backups. The default is 1 writer thread. The minimum is 1 and the

maximum is 99. This parameter cannot be specified for other RMU commands which accept the /LIBRARIAN qualifier for write operations such as RMU/BACKUP/AFTER_JOURNAL/LIBRARIAN since these commands only allow 1 writer thread which creates 1 database stream. This value will be set equal to the number of database storage areas if it exceeds that number.

- READER_THREADS=#
  Use # reader threads to read all the backup data streams from the LIBRARIAN created for the backup filename. The streams will be named BACKUP_FILENAME.EXT, BACKUP_FILENAME.EXT02, BACKUP_FILENAME.EXT03, up to BACKUP_FILENAME.EXT99. BACKUP_FILENAME.EXT is the backup file name specified in the RMU command excluding any specified device, directory or version number. The default extension name is ".RBF". The "READER_THREADS" parameter can only be specified for database restores and dumps of databases stored by RMU in the LIBRARIAN. A reader thread value of 1 is used for all other RMU commands that read data from the LIBRARIAN. The minimum READER_THREADS value is 1 and the maximum is 99. The default value is 1.
  The number of READER_THREADS for a database restore from the LIBRARIAN should be equal to or less than the number of WRITER_THREADS specified for the database backup or the number of reader threads will be set by RMU to be equal to the number of data streams actually stored in the LIBRARIAN by the backup. If the READER_THREADS specified for the restore are less than the WRITER_THREADS specified for the backup, RMU will partition the data streams among the specified reader threads so that all data streams representing the database are restored. Therefore, each reader thread may read more than one data stream.
- TRACE_FILE=file_specification
  The LIBRARIAN application which supports the MEDIA MANAGEMENT API V2.0 will write trace data to this file, if specified, as defined in the MEDIA MANAGEMENT API V2.0 specification.
- LEVEL_TRACE=#
  The level number of the trace data written by the LIBRARIAN application which supports the MEDIA MANAGEMENT API V2.0 as defined in the MEDIA MANAGEMENT API V2.0 specification (levels 0 through 2) or a higher level as defined by the LIBRARIAN application. Level 0 (trace all error conditions) is the default.
- LOGICAL_NAMES=(logical_name=equivalence_value,...)
  This parameter allows the user to specify a list of process logical names which the LIBRARIAN application may use to specify particular catalogs or archives for storing or retrieving backup files, or LIBRARIAN debug logical names, etc. See the LIBRARIAN specific documentation for the definition of these logical names. The list of process logical names will be defined by RMU prior to the start of the backup or restore operation.

### 7.3.13.7 Logical Names To Access LIBRARIAN Application

The following VMS logical names are for use with a LIBRARIAN application. These logical names need to be defined before the RMU backup or restore command is executed and should not be specified with the list of logical names specified with the /LIBRARIAN qualifier.

- RMU$LIBRARIAN_PATH
  This logical name must be defined to the file specification for the sharable LIBRARIAN image to be loaded and called by RMU backup and restore operations. The translation must include the file type ( ".EXE" for example) and must not include a version number. The shareable LIBRARIAN shareable image referenced must be an installed ( "known") image.

See the LIBRARIAN implementation documentation for the name and location of this image and how it should be installed. For a parallel RMU backup, RMU$LIBRARIAN_PATH should be defined as a system−wide logical name so that the multiple processes created by a parallel backup can all translate the logical.

```
$ DEFINE /SYSTEM /EXECUTIVE_MODE -
    RMU$LIBRARIAN_PATH librarian_shareable_image.exe
```

♦ RMU$DEBUG_SBT
This logical name is not required. If defined to any value, RMU will display debug tracing information messages from modules that make calls to the LIBRARIAN shareable image. This information may be helpful for support analysts from Oracle or your librarian vendor when analyzing problems. See the LIBRARIAN documentation for any other logical names or setup procedures specific to the particular LIBRARIAN implementation. For a parallel backup, RMU$DEBUG_SBT should be defined as a system logical so that the multiple processes created by a parallel backup can all translate the logical.

### 7.3.13.8 SQL/Services Required for RMU Parallel Backup

Oracle Rdb V7.1 SQL/Services is required for RMU parallel backup operations. However, no special changes are required to SQL/Services specific to RMU parallel backup operations to the LIBRARIAN. The LIBRARIAN should be installed and available on all nodes on which the parallel backup operation executes. As long as non−LIBRARIAN Rdb V7.1 parallel backup operations are currently working, no LIBRARIAN specific changes to the SQL/Services setup should be needed.

### 7.3.13.9 Listing and Deleting Data Streams

The RMU /LIBRARIAN command enables the user to list or delete data streams stored in the LIBRARIAN implementation based on the backup file name used for the RMU backup. The LIST and REMOVE options cannot be used together in the same RMU/LIBRARIAN command.

```
RMU /LIBRARIAN /LIST=(OUTPUT=disk:[directory]listfile.ext) FILENAME.RBF
RMU /LIBRARIAN /REMOVE=([NO]CONFIRM) FILENAME.RBF
```

FILENAME.RBF is the backup filename. Any device, directory or version number specified with the backup file name will be ignored. The backup file name must be the same name previously used for an RMU backup to the LIBRARIAN. A default file type of ".RBF" is assumed if none is specified.

The following command qualifiers are supported:

♦ /LIST=(OUTPUT=disk:[directory]listfile.ext)
"/LIST" used alone will display to the default output device. If the "OUTPUT" option is used, output will be displayed to the specified file. All data streams existing in the LIBRARIAN that were generated for the specified backup name will be listed. The information listed for each data stream name include:
    ◊ The backup stream name based on the backup file.
    ◊ Any comment associated with the backup stream name.
    ◊ The creation method associated with the backup stream name. This will always be STREAM to indicate creation by a backup operation.
    ◊ The creation date and time when the stream was backed up to the LIBRARIAN.
    ◊ Any expiration data and time specified for deletion of the stream by the LIBRARIAN.

- ◊ The media sharing mode which indicates if the media can be accessed concurrently or not. This is usually the case for disks but not tapes.
- ◊ The file ordering mode which indicates if files on the media can be accessed in random order or sequential order.
- ◊ Any volume label(s) for the media which contain the backup stream.

---

Implementation Specific

***Not all of these items will be listed depending on the particular LIBRARIAN implementation.***

---

- ♦ /REMOVE=([NO]CONFIRM)
  "/REMOVE" deletes all data streams existing in the LIBRARIAN that were generated for the specified backup name. This command should be used with caution. The user should be sure that a more recent backup for the database exists in the LIBRARIAN under another name before using this command. The "CONFIRM" option is the default. It will prompt the user to confirm that he wants to delete the backup from the LIBRARIAN. The user can then reply "Y(ES)" to do the deletion or "N(O)" to exit the command without doing the deletion if he wants to confirm that a more recent backup for the database exists in the LIBRARIAN that was generated using a different backup name. The user must specify the "NOCONFIRM" option if he does not want to be prompted. In this case, the deletion will be done with no confirmation prompt.
  The following additional optional keywords can be specified with either the /LIST qualifier or the /REMOVE qualifier. They must be specified and have no defaults. These are the same options discussed earlier for the /LIBRARIAN qualifier used with other RMU commands such as /BACKUP and /RESTORE.
    - ◊ TRACE_FILE=file_specification
      The LIBRARIAN application which supports the MEDIA MANAGEMENT API V2.0 will write trace data to this file, if specified, as defined in the MEDIA MANAGEMENT API V2.0 specification.
    - ◊ LEVEL_TRACE=n
      The level number of the trace data written by the LIBRARIAN application which supports the MEDIA MANAGEMENT API V2.0 as defined in the MEDIA MANAGEMENT API V2.0 specification (levels 0 through 2) or a higher level as defined by the LIBRARIAN application. Level 0 (trace all error conditions) is the default.
    - ◊ LOGICAL_NAMES=(logical_name=equivalence_value,...)
      This parameter allows the user to specify a list of process logical names which the LIBRARIAN application may use to specify particular catalogs or archives for listing or removing backup files, or LIBRARIAN debug logical names, etc. See the LIBRARIAN specific documentation for the definition of these logical names. The list of process logical names will be defined by RMU prior to the start of the list or remove operation.

## 7.3.14 Sanity Checks Added to RMU /VERIFY to Check TSNs and CSNs

Bug 2551131

Checks have been added to RMU /VERIFY to ascertain if the Transaction Sequence Numbers (TSNs) and Commit Sequence Numbers (CSN) have acceptable values. This verification is performed while verifying the root.

Three sanity checks have been added to achieve this. They are:

- Highest Active TSN is less than the TSN that will be assigned to the next transaction. If this check fails the following kind of error message is displayed.

  ```
  %RMU−E−HIGHTSNINV, Highest active TSN (1024:1024) is higher than
  the TSN that will be assigned next (0:256).
  ```
- Last Committed TSN is less than the TSN that will be assigned to the next transaction. If this check fails the following kind of error message is displayed.

  ```
  %RMU−E−LASTCMTSNINV, Last Committed TSN (1024:1024) is higher than
  the TSN that will be assigned next (0:256).
  ```
- Highest CSN is less than the next CSN that will be assigned. If this check fails the following kind of error message is displayed.

  ```
  %RMU−E−HIGHCSNINV, Highest CSN (1024:1024) is higher than the CSN
  that will be  assigned next (0:256).
  ```

These checks will be available starting with Oracle Rdb Release 7.1.2.

## 7.3.15 RMU /CLOSE /WAIT /NOCLUSTER Now Allowed

Bug 976101

In the past, if the /WAIT qualifier was used with ***RMU /CLOSE*** then it implied ***/CLUSTER***. Specifying ***/NOCLUSTER*** was not allowed. This restriction has been lifted. It is now possible to specify ***/NOCLUSTER*** in conjunction with the /WAIT qualifier. This provides the ability to have database shutdown complete on the local node before RMU returns to the DCL prompt. Specifying the /WAIT qualifier without the ***/NOCLUSTER*** qualifier will still imply ***/CLUSTER***, as it has in prior releases.

## 7.3.16 Native 64−bit Virtual Addressing for Row Caches

Oracle Rdb Release 7.1.2 provides enhancements to the Row Cache feature to utilize the native 64−bit virtual memory addressing capabilities of the Alpha processor and OpenVMS. Utilizing these enhancements, applications are now able to more easily access row caches with significantly larger numbers of records being cached.

### 7.3.16.1 Background

Within Oracle Rdb, the VLM (Very Large Memory, or "LARGE MEMORY IS ENABLED") feature was created circa 1995 for Rdb release 7.0 to allow access to more than 32 bits worth of virtual address space (the traditional VMS address space size). This interface was implemented because, at the time, programs on OpenVMS Alpha did not have the ability to directly access memory outside a 32−bit virtual address space.

In addition, the "SHARED MEMORY IS SYSTEM" feature was implimented to help reduce consumption of process virtual address space in the "program region" (P0 region). By moving database shared memory sections from P0 to "system" (S0/S1) address space, additional program code and buffers could be stored in P0 address space. However, even this additional virtual address space was limited to something less than 2 GB (gigabytes) of usable memory.

Starting with OpenVMS Alpha V7.0, the operating system provides native support for a 64−bit virtual address space, defined by the Alpha architecture. This capability makes 64−bit virtual address space available to applications. OpenVMS and Current Alpha architecture implementations support 8 TB (terabytes) of virtual address space.

Previously, the Oracle Rdb Row Cache feature was limited to something less than 33 million total cached rows per database. This limitation was due primarily to storing row cache related data structures in 32−bit address space. Even with the "LARGE MEMORY IS ENABLED" attribute, some data structures had to be located in 32−bit virtual address space and lead to this restriction.

## 7.3.16.2 64−bit Addressing

Starting with Oracle Rdb Release 7.1.2, the Row Cache feature utilizes the native 64−bit virtual addressing support within the Alpha processor and OpenVMS. Various data structures related to the Row Cache feature are now created in the OpenVMS "P2" 64−bit virtual address space. The existing Row Cache memory mapping features "LARGE MEMORY IS ENABLED" and "SHARED MEMORY IS SYSTEM" have been eliminated and replaced with this native 64−bit virtual addressing. Row caches on Alpha processors are now always mapped in 64−bit process virtual address space.

## 7.3.16.3 No Application or Database Changes Required

There should be no user or application visible effects due to the Oracle Rdb implementation of native 64−bit virtual addressing for Row Caches. If the "RESIDENT" attribute is specified for a row cache, the cache will be created as a memory−resident global section utilizing OpenVMS "shared page tables" (potentially with granularity hints). The OpenVMS "shared page tables" capability for memory−resident global sections can help reduce physical memory consumption by allowing multiple processes to share page table entries for the global section. For very large global section, "granularity hints" allow ranges of pages to be mapped by a single translation buffer entry within the Alpha processor leading to improved translation buffer hit rates.

When the shared memory section for a row cache is to be memory−resident, the pages for the section are always resident in physical memory. The pages are not placed into the process' working set list when the process maps to the global section. The pages are also not charged against the process' working set quota or against any page−file quota. Pages within memory−resident global sections are not backed by the pagefile or by any other file on disk. The user must have the rights identifier VMS$MEM_RESIDENT_USER to create the memory−resident global section.

## 7.3.16.4 Deprecated Attributes

The row cache memory mapping features "LARGE MEMORY IS ENABLED" and "SHARED MEMORY IS SYSTEM" have been replaced with the "RESIDENT" attribute. If the "LARGE MEMORY IS ENABLED" or "SHARED MEMORY IS SYSTEM" attributes are specified for a row cache, the cache is considered to be set "RESIDENT". Though deprecated, the "LARGE MEMORY IS ENABLED" and "SHARED MEMORY IS SYSTEM" attributes are internally considered as synonyms for "RESIDENT".

The following Oracle Rdb Row Cache attributes have become deprecated in SQL:

- "LARGE MEMORY IS ENABLED"
- "SHARED MEMORY IS SYSTEM"
- "WINDOW COUNT"

The following Oracle Rdb Row Cache attribute keywords have become deprecated with the "RMU /SET ROW_CACHE /ALTER" command:

- WINDOW_COUNT=n
- SHARED_MEMORY=TYPE=SYSTEM

The "LARGE MEMORY IS ENABLED" and "SHARED MEMORY IS SYSTEM" attributes remain supported and functional for database and global buffer shared memory configuration.

#### 7.3.16.5 Cache Size Limits

In this release of Oracle Rdb, the total number of rows for any individual cache (the combination of "live" rows and snapshot rows) is limited to 2,147,483,647. This restriction may be relaxed in a future Oracle Rdb release.

#### 7.3.16.6 Row Cache Feature Only

This change to utilize native 64−bit virtual addressing is specific to the Row Cache feature in this release of Oracle Rdb. Rdb database global buffers, for example, continue to be mapped into 32−bit virtual address space and continue to support the "LARGE MEMORY IS ENABLED" and "SHARED MEMORY IS SYSTEM" attributes. No additional support for 64−bit virtual addressing (including via callable interfaces such as SQL) is provided in this release of Oracle Rdb. Oracle is considering additional possible uses for native 64−bit virtual addressing within Oracle Rdb for future releases.

#### 7.3.16.7 System Parameters

Shared memory sections using the "LARGE MEMORY IS ENABLED" or "SHARED MEMORY IS SYSTEM" features were previously not created as OpenVMS global sections and were not directly effected by the global section system parameters (specifically GBLSECTIONS, GBLPAGES and GBLPAGFIL). Because all row cache shared memory sections are now global sections on OpenVMS, it is possible that the global section system parameters may have to be increased in order to map large caches that previously relied on the "LARGE MEMORY IS ENABLED" or "SHARED MEMORY IS SYSTEM" features.

---

Reserved Memory Registry

***The Reserved Memory Registry allows an OpenVMS system to be configured with large amounts of memory set aside for use within memory−resident sections or other privileged code. This release of Oracle Rdb does not support use of the OpenVMS Reserved Memory Registry for registering Oracle Rdb shared memory sections. This restriction may be relaxed in a future Oracle Rdb release.***

---

### 7.3.16.8 Additional Information

Refer to the following documents for additional information about Alpha processors and OpenVMS addressing and memory management:

- OpenVMS Programming Concepts Manual
- OpenVMS System Services Reference Manual
- OpenVMS Calling Standard
- OpenVMS System Manager's Manual
- OpenVMS Alpha Partitioning and Galaxy Guide
- Alpha Architecture Handbook

## 7.3.17 Snapshots In Row Cache

Oracle Rdb Release 7.1.2 provides enhancements to the Row Cache feature to allow snapshot copies of rows to be stored in Row Cache shared memory rather than in the on−disk snapshot storage areas. Utilizing these enhancements, applications are now able to more easily access and modify rows with reduced database I/O and locking operations. These features are enabled and configured on a per−cache basis.

### 7.3.17.1 Background

A row cache is a section of globally accessible memory that contains copies of rows. Row caching provides the ability to store frequently accessed rows in memory, reducing disk I/O. The rows remain in memory even when the associated page has been flushed back to disk. A row cache can contain index structures as well as table data.

The snapshot mechanism in Oracle Rdb allows read−only transactions to see a consistent view of the database while other transactions update the database. The previous versions of rows are written to special snapshot areas of the database by the transactions that update the rows.

By default, snapshot copies of rows are stored in database snapshot storage area files. The "Snapshots in Row Cache" feature allows snapshot rows to be stored in a designated section of shared memory. With this enhancement, read−only transactions can quickly read snapshot copies of rows from memory and read−write transactions can quickly write snapshots. The reading and writing of the snapshot information can be accomplished with no database page I/O or associated database page locking.

### 7.3.17.2 Configuration

Each defined row cache for a database can be designated to allow a specified number of snapshot rows to be stored in the cache. The number of snapshot rows allowed is specified in addition to the number of database rows that can be stored in the cache. Because many versions of a row may be stored in the snapshot portion of the cache, the number of snapshot rows and cache rows are specified independently.

As the snapshot portion of the row cache is effectively an extension of the row cache itself, most attributes of the cache are shared with the snapshot portion. These attributes include:

- The maximum row size (as is specified with the ROW LENGTH is n BYTES parameter)

- Memory location specification (SHARED MEMORY IS PROCESS, SHARED MEMORY IS SYSTEM, LARGE MEMORY, and so on)
- Allocate Set count

### 7.3.17.3 Space Reclaimation

A snapshot copy of a database record must be maintained until there are no transactions in the database older than the transaction that stored the snapshot copy of the record. As the oldest transactions in the database commit, space used to store snapshot records (either in the snapshot storage areas or in the snapshot portion of a cache) may be re−used for storing newer snapshots. By keeping transactions relatively short, the number of snapshot rows that need to be stored can be reduced.

### 7.3.17.4 Objects in Mixed Format Areas

With this release of the "Snapshots in Row Cache" feature, only objects (index nodes and data records) stored in uniform−format storage areas utilize the ability to store snapshots in a row cache. Objects stored in mixed format storage areas are unable to have snapshot copies stored in a row cache.

This restriction results from internal mechanisms used to perform sequential scans in the database and interactions with the retrieval of snapshot information. This restriction is expected to be relaxed in a future Oracle Rdb release.

### 7.3.17.5 SQL Syntax

The SQL "ALTER DATABASE ... ALTER ROW CACHE", "ALTER DATABASE ... ADD ROW CACHE", and " CREATE DATABASE ... CREATE ROW CACHE" commands can be used to set parameters for the snapshot portion of a row cache. The syntax to support snapshots in cache is "ROW SNAPSHOT [ IS ] { DISABLED | { ENABLED [ ( CACHE SIZE IS N ROWS ) ] }".

- The "ROW SNAPSHOT IS DISABLED" option disables storing snapshot copies of rows within the cache.
- The "ROW SNAPSHOT IS ENABLED (CACHE SIZE IS n ROWS)" option enables storage of snapshot copies of rows within the cache and specifies the number of snapshot "slots" to allocate for the cache.

If you do not specify the CACHE SIZE clause for the ROW SNAPSHOT IS ENABLED option, Oracle Rdb creates a cache that can contain up to 1000 snapshot rows.

The following example demonstrates using SQL to modify the "C1" cache to disable storage of snapshot rows in cache and to modify the "C5" cache to enable storage of snapshot rows in the cache with a snapshot cache size of 12345 rows:

```
SQL> ALTER DATABASE FILE X$
cont> ALTER CACHE C1
cont>  ROW SNAPSHOT IS DISABLED;
SQL> ALTER DATABASE FILE X$
cont> ALTER CACHE C5
cont>  ROW SNAPSHOT IS ENABLED (CACHE SIZE IS 12345 ROWS);
```

### 7.3.17.6 RMU Syntax

The "RMU /SET ROW_CACHE" command can be used to set parameters for the snapshot portion of a row cache. The "/ALTER=(...)" qualifier accepts a "SNAPSHOT_SLOT_COUNT=n" keyword. The value specified on the SNAPSHOT_SLOT_COUNT keyword sets the number of snapshot slots in the cache. A value of zero disables the snapshot portion for the specified cache.

The following example modifies the database MYDB to set the snapshot slot count for the cache "EMPL_IDX" to 25000 slots and disables snapshots in cache for the "SALES" cache:

```
$ RMU /SET ROW_CACHE DGA0:[DB]MYDB.RDB -
  /ALTER=(NAME=EMPL_IDX, SNAPSHOT_SLOT_COUNT=25000) -
  /ALTER=(NAME=SALES, SNAPSHOT_SLOT_COUNT=0)
```

### 7.3.17.7 Snapshot Cache Sizing

Because the application and workload behaviour determine the number of database rows that are modified and the transaction length, it is not reasonable to make specific recommendations for sizing the snapshot portion of caches for all application and database types. The ratio of the size of the snapshot cache to the main cache may be similar to the ratio of the database snapshot storage area to the live storage area.

The snapshot portion of a row cache may be larger (may contain more rows) than the "main" row cache itself. The snapshot portion of a row cache may also be much smaller.

If an application has long running transactions and active read−write transactions modifying data, many snapshot copies of the modified data may need to be maintained. This can require caches with many snapshot rows for those caches with heavy update activity.

When the snapshot portion of a cache fills and no slots are available for re−use (due to the age of the oldest transaction in the database), read−write transactions may need to "overflow" snapshot records from cache to disk. This overflow operation can be quite costly in terms of CPU time and disk I/O operations. When a snapshot cache is discovered to be full and a read−write transaction must store a new snapshot copy of a row, all existing snapshot copies for that row must be written from the cache to the snapshot storage area on disk. And all future snapshot operations to the cache must also be written to disk.

When the snapshot portion of a row cache is marked as being "full", the row cache server (RCS) process periodically checks the cache to see if space is available for reuse. Similar to the algorithms governing space recollection in snapshot storage areas, this space only becomes available when the oldest transaction in the database commits. When the RCS process finds reclaimable space in the snapshot portion of a cache that is marked "full", it will clear the "full" indicator to permit new snapshot copies to be stored in the cache by read−write transactions.

### 7.3.17.8 Performance and Operational Considerations

When a row is removed from the cache (due to it becoming fragmented, growing too large for the cache, or from a TRUNCATE TABLE operation), all snapshot copies for the row must be written from cache back to the snapshot storage area on disk. This can be a relatively costly and slow operation. This can usually be avoided by insuring that caches are sized with slots large enough for the data being stored.

At certain times during normal database operations, all modified rows must be written from cache(s) back to the physical database storage areas. Events that require writing all modified data back to the database include:

- Database close
- RMU /VERIFY
- RMU /BACKUP

Prior to "Snapshots in Row Cache", it was unlikely that very many modified rows would remain in cache memory when database snapshots were enabled. Now, for those caches configured with snapshots in cache, the cache itself may have many more modified rows. When there are many modified rows in memory, it may take a significant amount of time to write all of these rows back to the database. You may need to plan based on the amount of time required to, for example, initiate backup operations, if there may be a large number of modified rows in cache.

By removing delays introduced by disk I/O operations, applications may tend to experience improved performance. Some systems, however, may see a significant reduction in system idle time due to the reduction in I/O waiting. Presumably, this will be reflected in an increase of overall application performance as the computer system is now being more effectively utilized.

## 7.3.17.9 Statistics

The "Row Cache Status" display of the RMU/SHOW STATISTICS utility provides information about the state of a single row cache and now includes status information about the snapshot portion of the cache.

```
Node: CLICK (2/2/2) Oracle Rdb      Perf. Monitor 17−FEB−2003 22:20:39.61
Rate: 3.00 Seconds          Row Cache Status        Elapsed: 00:00:31.79
Page: 1 of 1        DISK$DEMO1:[RDBDEMO]OLTP.RDB;1          Mode: Online
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
                             For Cache: TRD_IDX1
Statistic.Name Stat.Value Percent

Total slots:       2000  100.0% Slot Length: 1000  Hash slots: 2048
Slots full:         876   43.8% Use:        225  25.6%
Slots empty:       1124   56.2% Rsv:        132  11.7%
Marked Slots:       347   17.3% Hot:        347 100.0% Cold:     0   0.0%
Clean Slots:       1653   82.6% Hot:          0   0.0% Cold:  1653 100.0%
Used Space:         876k  43.8% Wstd:        0k  0.0%
Free Space:        1124k  56.2%
Hash Que Lengths:  Empty:1245 1:730      2:73       3:0        4+:0
Cursor position:   1008 of 2000 wrapped 0 times
Cache latched:     No
Cache is full:     No          Cache modified:  Yes  Snapshot is full: No
Number of checkpoints: None
Cache Recovery:    0:3577
Snap Slots:         500  100.0% Ful:      399  79.8% Rcl:     393  78.6%
Snap Cursor:  45 of 500 (slot 2045) wrapped 2 times
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
```

The following fields provide information about the snapshot portion of the cache.

- Snapshot is full − If snapshots within cache are enabled for the row cache, indicates if the cache has been flagged as having no snapshot slots that can be used to store snapshot records until the oldest transaction in the database commits and allows snapshot slots to be "reclaimed" for re−use.

- Snap Slots − Indicates the number of snapshot slots configured.
- ...Ful − How many of the slots contain snapshot records.
- ...Rcl − How many of the slots contain snapshot records that can be "reclaimed" for re−use.
- Snap Cursor − Indicates the current cursor position within the snapshot portion of the cache; processes allocating slots in the cache start searching for available space at this cursor position.
- ...wrapped − How many times the entire snapshot portion of the cache has been scanned and the allocation cursor was reset to the beginning of the cache.

In this example display, the row cache itself is configured with 2000 slots and the snapshot portion of the cache is configured for 500 slots (the snapshot portion of the cache can be configured to be larger or smaller than the cache itself). The current "Snap Cursor" position is at slot 45 within the 500 snapshot slots.

Within the RMU/SHOW STATISTICS utility, the "Zoom" sub−screen of the "Hot Row Information" display can be used to examine the actual in−cache contents of a row. On this display, the sign (positive or negative) of the "SnapPage" value indicates if the snapshot pointer references a page on disk in the snapshot storage area (a positive value) or a slot number within the snapshot portion of the cache (a negative value). For example, the following display shows a "Zoom" sub−screen for the record in cache with a database key of 56:662:0.

```
Node: MARVEL (1/1/1)      Oracle Rdb  Perf. Monitor  3−FEB−2003 23:11:06.27
Rate: 3.00 Seconds          Hot Row Information        Elapsed: 04:14:58.28
Page: 1 of 6           DGA127:[T]MF_PERSONNEL.RDB;587          Mode: Online
−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−
                            For Cache: N1 (unsorted)
Area:Page:Ln #Users State Length SlotNo Area:Page:Ln #Users State Length
Empty             0            0      0 Empty             0           0

 +−− DBK=56:662:0 LEN=17 TSN=0:132 SnapPage=−101 VNO=2 −−−−−−−−−−−−−−−−−+
 |                                                                      |
 |                     001E  0000  line 0 (56:662:0) record type 30   |
 |                  00 0001  0002  1 byte in 0 sets/dynamic items     |
 |                           ....  12 bytes of static data            |
 |   FC0000820200008201000102  0005   data '...........ü'             |
 |                                                                      |
 +−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−−+
```

In this example, the snapshot pointer is −101 indicating, because it is negative, that the first entry in the snapshot chain can be found in snapshot slot 101 within the cache. Note that a snapshot pointer of −1 indicates the end of a snapshot chain in this context.

The RMU /DUMP /ROW_CACHE command can also be used to format and display the in−memory cache contents for a row cache of an open database. In this display output, negative snapshot pointer values also indicate snapshot pointers within the row cache.

## 7.3.17.10 Importance of the After−Image Journal

Any time the Oracle Rdb "Fast Commit" feature is utilized, the after−image journal (AIJ) is the *only* place where changed database records are known to be written to persistent storage when a transaction commits.

Protecting the after−image journal file(s) is thus very important. Oracle encourages use of data protection features such as disk volume shadowing and especially the Oracle Rdb Hot Standby feature to help ensure the safety of the contents of the AIJ.

The Row Cache "backing store" (also known as .RDC) files are also important to ensure rapid database recovery after a system failure. These files contain the modified row content of caches as of the most recent row cache server (RCS) checkpoint operation. Oracle encourages use of data protection features such as disk volume shadowing to help ensure the safety of the contents of the "backing store" files.

## 7.3.18 Performance Enhancements for RMU /RECOVER with Optimized After−Image Journals

Several enhancements and performance improvements have been made to the creation and processing of optimized after−image journal files. These changes should result in a significant reduction in elapsed time when using optimized after−image journals for recovery.

The RMU /OPTIMIZE /AFTER_JOURNAL command optimizes a backed up after−image journal (.AIJ) file for database recovery (rollforward) operations by eliminating unneeded and duplicate journal records, and by ordering journal records. An optimized after−image journal file created by the RMU /OPTIMIZE /AFTER_JOURNAL command can provide better recovery performance for your database than a non−optimized after−image journal. A potential benefit of this improved recovery performance is that the database is made available to users sooner.

By default, the RMU /OPTIMIZE /AFTER_JOURNAL command orders the after−image journal records by ascending physical DBKEY. The order of records in an optimized AIJ file determines the sequence that pages are accessed by a subsequent RMU /RECOVER command. Sorting AIJ records by physical DBKEY can improve I/O performance at recovery time by reducing disk head motion. However, an ascending physical DBKEY sequence also causes the database to be recovered sequentially, one storage area at a time. This typically results in only one disk device being accessed at a time, often with sequential disk read and write operations.

Significant enhancements have been made to the optimization and usage of optimized after−image journals. These changes include the "/RECOVERY_METHOD" qualifier for the RMU /OPTIMIZE /AFTER_JOURNAL command to allow an alternate record ordering to be used, and asynchronous read−ahead and write−behind for database access during recovery using an optimized after−image journal.

The "/RECOVERY_METHOD" qualifier for the RMU /OPTIMIZE /AFTER_JOURNAL command allows two possible order types:

- SEQUENTIAL − AIJ records are ordered by physical DBKEY in a AREA:PAGE:LINE sequence. This is the traditional method used by the RMU /OPTIMIZE /AFTER_JOURNAL command and is the default.
- SCATTER − AIJ records are ordered by a sort key of PAGE:AREA:LINE (page number, area number and line number). This order often allows the RMU /RECOVER command to perform much more effective I/O prefetching and writing to multiple storage areas simultaneously (typically where storage areas of the database are distributed among multiple disk devices).

SCATTER ordering tends to allow more disk devices to be active during the recovery process. This, in turn, should help reduce idle CPU time and allows the recovery to complete in less time. However, because database configurations vary widely, Oracle recommends that you perform tests with both SCATTER and SEQUENTIAL ordering of the optimized after−image journals to determine which method produces the best results for your system.

Note that because an optimized AIJ file is not functionally equivalent to the original AIJ file, the original AIJ file should not be discarded after it has been optimized.

You cannot use optimized AIJ files with the following types of recovery operations:

- By−area recovery operations (recovery operations that use the RMU Recover command with the Areas qualifier).
- By−page recovery operations (recovery operations that use the RMU Recover command with the Just_Corrupt qualifier).
- RMU Recover commands with the Until qualifier. The optimized AIJ does not retain enough of the information from the original AIJ for such an operation.
- Recovery operation where the database or any storage areas (or both) are inconsistent with the optimized AIJ file. A database or storage area will be inconsistent with the optimized AIJ file if the transaction sequence number (TSN) of the last committed transaction of the database or storage area is not equal to the TSN of the last committed transaction in the open record of the AIJ file. The last committed TSN in the optimized file represents the last transaction committed to the database at the time the original AIJ file was created.

As a workaround for these restrictions against using optimized AIJ files in these recovery operations, use the original, unoptimized AIJ files in these situations instead. Oracle recommends that you do not discard the original AIJ file after it has been optimized.

When using an optimized after−image journal for recovery, the optimal number of buffers specified with the "/AIJ_BUFFERS" qualifier depends on the number of "active" storage areas being recovered. For those journals optimized with "/RECOVERY_METHOD=SEQUENTIAL" (the default), a buffer count of perhaps 250 to 500 is usually sufficient.

When using journals optimized with "/RECOVERY_METHOD=SCATTER", reasonable performance can usually be attained with a buffer count of about five times the number of "active" storage areas being recovered (with a minimum of perhaps 250 to 500 buffers).

"Active" storage areas refers to those areas that have records with modifications reflected in the optimized after−image journal. If only a small number of storage areas are actively modified, less recovery buffers may be needed. The CPU cost of using "excessive" numbers of buffers tends to be relatively small.

When using non−optimized after−image journals for recovery, the RMU /DUMP /AFTER_JOURNAL command can be used to suggest an optimal number of recovery buffers. In effectively all cases though, the default of 20 buffers is probably not sufficient for best performance. Oracle suggests specifying a buffer count of at least 5000 for most databases as a reasonable starting point.

The number of asynchronous prefetch (APF) buffers is also a performance factor during recovery. For recovery operations of optimized after−image journals, the RMU /RECOVER command sets the number of APF buffers (also known as the APF "depth") based on the values of the process quotas ASTLM, BYTLM and the specified "/AIJ_BUFFERS" value. Specifically, the APF depth is set to the maximum of:

- ♦ 50% of the ASTLM process quota
- ♦ 50% of the DIOLM process quota
- ♦ 25% of the specified "/AIJ_BUFFERS" value

Further, accounts and processes that perform RMU /RECOVER operations should be reviewed to ensure that various quotas are set to ensure high levels of I/O performance. Table 7−2 lists suggested quota values for recovery performance.

***Table 7−2 Recommended Minimum Process Quotas***

| Quota | Setting |
|---|---|
| DIOLM | Equal to or greater than half of the count of database buffers specified by the "/AIJ_BUFFERS" qualifier. Minimum of 250. |
| BIOLM | Equal to or greater than the setting of DIOLM. |
| ASTLM | Equal to or greater than 50 more than the setting of DIOLM. |
| BYTLM | Equal to or greater than 512 times the database buffer size times one half the value of database buffers specified by the "/AIJ_BUFFERS" qualifier. Based on a 12 block buffer size and the desire to have up to 100 asynchronous I/O requests outstanding (either reading or writing), the minimum suggested value is 614,400 for a buffer count of 200. |
| WSQUOTA, WSEXTENT | Large enough to avoid excessive page faulting |
| FILLM | 50 more than the count of database storage areas and snapshot storage areas. |

The RMU /DUMP /AFTER_JOURNAL command indicates the type of optimization (sequential or scattered) when dumping an optimized after−image journal file. The first record in the after−image journal is an "Open" record and contains an indication that the journal is optimized and what type of optimization was specified as shown in the following example (the line reading "Type is Optimized"):

```
1/1   TYPE=O, LENGTH=510, TAD=27−MAY−2003 08:25:31.67, CSM=00
    Database DPA86:[AIJOPT]MF_PERSONNEL.RDB;1
    Database timestamp is 10−DEC−1996 10:17:31.13
    Facility is "RDMSAIJ ", Version is 711.1
    Database version is 71.0
    AIJ Sequence Number is 17231
    Last Commit TSN is 0:1384096
    Synchronization TSN is 0:0
    Journal created on VMS platform
    Type is Optimized (Scatter)
    Open mode is Initial
    Journal was backed up on 27−MAY−2003 08:26:26.25
    Backup type is Quiet−Point
    I/O format is Block
    Commit−to−Journal optimization disabled
    Switchover by process 2023D558
    AIJ journal activation ID is 00A207B1F9111F6B
    LogMiner is enabled
```

## 7.3.19 Enhancements to INSERT ... FILENAME for LIST OF BYTE VARYING Data

Enhancement 2738471

The INSERT INTO CURSOR ... FILENAME statement loads the contents of the specified file into the LIST OF BYTE VARYING column. In prior versions, the user could specify BINARY or TEXT as the type of data being inserted. This release of Oracle Rdb V7.1 now includes a new type, CHARACTER VARYING.

- BINARY
  Used to load unformatted data such as images, audio files, etc. The contents are broken into 512 octet segments during INSERT.
- TEXT
  Used to load text, a terminator is added to each segment loaded. The contents are written one line to a segment with trailing terminators carriage return (CR) and line feed (LF).
- CHARACTER VARYING
  Used to load text but with no terminator. The contents are written one line to a segment.

In addition, this release allows the TEXT and CHARACTER VARYING source to contain segments of up to 65500 bytes in length. In prior releases, the upper limit was 512 octets.

Interactive SQL now also reports the number of segments inserted and the length of the longest segment. To disable this output, use the SET DISPLAY NO ROW COUNT statement.

The following example shows a sample session that inserts a large text file into a single LIST OF BYTE VARYING column.

```
SQL> create table samples (a list of byte varying);
SQL>
SQL> declare a insert only table cursor for select a from samples;
SQL> declare b insert only list cursor for select a where current of a;
SQL>
SQL> open a;
SQL> insert into cursor a default values;
1 row inserted
SQL>
SQL> open b;
SQL> insert into cursor b
cont>   filename 'WEEKLY_REPORT.DAT' as character varying;
47706 segments inserted (maximum length 270)
SQL> close b;
SQL>
SQL> close a;
```

This statement can only be used in interactive SQL and dynamic SQL.

## 7.3.20 Default for RMU CRC Qualifier Changed to /CRC = AUTODIN_II

The default behavior for the CRC qualifier for the following RMU commands is changed in Oracle Rdb V7.1.2:

- Backup
- Backup After_Journal
- Backup Plan
- Optimize After_Journal

The default value for the CRC qualifier on the above backup commands will be CRC = AUTODIN_II.

Oracle Corporation recommends that you accept the new behavior for your applications. The new default behavior prevents undetected corruption in backup media. /CRC specifies that software CRC checking code is to be computed and stored in the data blocks of the output. This level of software checking provides full end−to−end data consistency checks.

- /CRC = AUTODIN_II − Uses the AUTODIN−II polynomial for the 32−bit cyclic redundancy check (CRC) calculation and provides the most reliable end−to−end error detection. Typing /CRC is sufficient to select the /CRC=AUTODIN_II qualifier. It is not necessary to type the entire qualifier.
- /CRC = CHECKSUM − Uses one's complement addition, which is the same computation used to do a checksum of the database pages on disk.
- /NOCRC − Disables end−to−end error detection.

---

Use of /CRC

***Please note that the use of /CRC on the AIJ backup command has meaning when used in conjunction with the /FORMAT=NEW_TAPE qualifier. For more information, see the RMU Help on /BACKUP/AFTER/FORMAT.***

---

## 7.3.21 Index Estimation

Predicate estimation is being used increasingly in the Rdb optimizer to determine the cost and productivity of various index scans.

When a particular query is executed, the conditions in the record select expression, the "where" clause of an SQL statement, determine which rows will be selected. These conditions, or predicates, can be used to limit the parts of an index that are scanned to find data records.

Consider the following SQL query.

```
SQL> SELECT * FROM employees WHERE last_name='Toliver'
cont>  and first_name='Alvin';
```

If there were an index on first_name and a second index on last_name, then Rdb would have to decide which of the two indices would be most efficient for retrieving the data. To do this, Rdb examines each index to find out roughly how many rows would be found through that index. For example, how many 'Toliver' rows would be found in the last_name index.

Historically, Rdb uses estimation in the dynamic optimizer. The dynamic optimizer uses estimation to calculate costs of index scans on competing indices. This information is used to ensure the most efficient indices are scanned first.

During request compilation, the Rdb optimizer uses the selectivity of each expression to help cost various retrieval strategies to determine the most efficient method for retrieving the data.

Where an expression compares a literal value (e.g. WHERE field1=42), and an index exists on that field, the static optimizer can use estimation to obtain from the actual data an estimate for that predicate. In other words, how many rows would actually have the value forty−two in the field called "field1"?

This feature is called ***Sampled Selectivity*** and is described in a Technical Journal article available on OTN and MetaLink.

### 7.3.21.1 How Estimation Is Performed

Normally, Rdb performs estimation on indexes of *TYPE IS SORTED* and *TYPE IS SORTED RANKED* by descending the index structure to locate the index node where the selected range spans more than one key value in the node. This is termed the split level.

The *REFINE_ESTIMATES* flag can be used to modify the behaviour of the estimation process. For a complete description of the estimation process, including the new features, please refer to the Rdb Technical Note available through MetaLink.

The *REFINE_ESTIMATES* flag does not change the behaviour of the estimation process for indexes of *TYPE IS SORTED.*

For ranked indexes of *TYPE IS SORTED RANKED*, the *REFINE_ESTIMATES* flag allows estimation to descend beyond the split level to obtain a far more accurate estimate and enables rules to be enforced for how far estimation should proceed.

In addition to the new functionality, the execution trace has been significantly enhanced. In particular, the use of *SET FLAGS 'EXECUTION,DETAIL(1)'* will display significantly more information about the estimation process.

In the following example, the indexes contain the unique values 1 to 100,000. Index I21 is *TYPE IS SORTED* and index I22 is *TYPE IS SORTED RANKED.*

A cursor is opened selecting a range of exactly two keys for two records. In the first open, the keys are such that the estimation descends all the way to the level one node and therefore correctly estimates the number of records from each index as two.

However, in the second open, the keys were chosen such that they happened to span a separator in the index root node. So even though the query would only find two rows in the index, the estimation was erroneously high.

```
SQL> set flags 'strategy,detail(1),exec'
SQL> declare :a,:b,:c,:d int;
SQL> begin
cont> set :a=1; set :b=2; set :c=1; set :d=2;
cont> end;
SQL> declare c1 cursor for select count(*) from t2
cont> where f1 between :a and :b
cont> and f2 between :c and :d;
SQL> open c1;
```

```
~S#0003
Tables:
  0 = T2
Aggregate: 0:COUNT (*)
Leaf#01 BgrOnly 0:T2 Card=100000
  Bool: (0.F1 >= <var0>) AND (0.F1 <= <var1>) AND (0.F2 >= <var2>) AND (0.F2 <=
        <var3>)
  BgrNdx1 I21 [1:1] Fan=17
    Keys: (0.F1 >= <var0>) AND (0.F1 <= <var1>)
  BgrNdx2 I22 [1:1] Fan=17
    Keys: (0.F2 >= <var2>) AND (0.F2 <= <var3>)
~Estim  Ndx1 Sorted: Split lev=1, Seps=2 Est=2 Precise
~Estim  Ndx2 Ranked: Nodes=1, Min=2, Est=2 Precise IO=3
~Estim  RLEAF Cardinality=  1.0000000E+05
~E#0003.01(1) Estim   Index/Estimate 1/2 2/2
~E#0003.01(1) BgrNdx1 EofData  DBKeys=2  Fetches=0+0  RecsOut=0 #Bufs=1
~E#0003.01(1) BgrNdx2 FtchLim  DBKeys=0  Fetches=0+0  RecsOut=0
~E#0003.01(1) Fin     Buf      DBKeys=2  Fetches=0+1  RecsOut=2
SQL> close c1;
SQL> begin
cont> set :a=35287; set :b=35288; set :c=6207; set :d=6208;
cont> end;
SQL> open c1;
~Estim  Ndx1 Sorted: Split lev=4, Seps=1 Est=5534
~Estim  Ndx2 Ranked: Nodes=309, Min=0, Est=6205 IO=0
~Estim  RLEAF Cardinality=  1.0000000E+05
~E#0003.01(2) Estim   Index/Estimate 1/5534 2/6205
~E#0003.01(2) BgrNdx1 EofData  DBKeys=2  Fetches=1+0  RecsOut=0 #Bufs=1
~E#0003.01(2) BgrNdx2 FtchLim  DBKeys=0  Fetches=0+0  RecsOut=0
~E#0003.01(2) Fin     Buf      DBKeys=2  Fetches=0+1  RecsOut=0
SQL> close c1;
```

The next example shows the difference in the ranked index estimate on I22 when refinement is enabled.

```
SQL> select count(*) from t2
cont> where f1 between :a and :b
cont> and f2 between :c and :d;
~S#0004
Tables:
  0 = T2
Aggregate: 0:COUNT (*)
Leaf#01 BgrOnly 0:T2 Card=100000
  Bool: (0.F1 >= <var0>) AND (0.F1 <= <var1>) AND (0.F2 >= <var2>) AND (0.F2 <=
        <var3>)
  BgrNdx1 I21 [1:1] Fan=17
    Keys: (0.F1 >= <var0>) AND (0.F1 <= <var1>)
  BgrNdx2 I22 [1:1] Fan=17
    Keys: (0.F2 >= <var2>) AND (0.F2 <= <var3>)
~Estim  Ndx1 Sorted: Split lev=4, Seps=1 Est=5534
~Estim  Ndx2 Ranked: Nodes=2, Min=2, Est=2 Precise IO=2
~Estim  RLEAF Cardinality=  1.0000000E+05
~E#0004.01(1) Estim   Index/Estimate 2/2 1/5534
```

Notice that, in this case, the estimate for index I22, which is background index 2, is now precisely two even though the previous query indicated the split level for the same range was very high in the index.

The ranked estimation is more accurate because estimation refinement descended the index beyond the split level until a reasonable estimate was obtained. The estimate for sorted index I21 is still inaccurate because estimation refinement is only available for ranked indices.

Refinement rules allow control of the estimation process by limiting the total IO based on the smallest estimate obtained for each execution of a request. If the first index estimates that 10 rows will be returned, the estimation process can be limited to 10 IO's, on the assumption that fetching the records should take not more than ten IO's.

The ranked index estimation refinement rules are:

- REFINE_ESTIMATES(1) − Use IO to limit the descend to split level.
- REFINE_ESTIMATES(2) − Use IO to limit the refinement, where we read beyond the split level.
- REFINE_ESTIMATES(4) − Limit refinement until the known true branches of an index contain more records than branches of the index that are not completely included in the range selected.
- REFINE_ESTIMATES(8) − As each node in the index is processed for estimation, the error in that estimate is calculated. This rule terminates estimation once the calculated error in the estimate is less than ten percent of the estimate.
- REFINE_ESTIMATES(16) − Enable refinement. If this rule is the only refinement rule enabled, refinement will attempt to obtain a precise estimate regardless of the cost. If other refinement rules are also enabled, those rules will be enforced and this rule is not required.

Estimation refinement rules can be combined by adding their values. In this way, all ranked estimation refinement rules can be enabled by using *SET FLAGS 'REFINE_ESTIMATES(15)'*. The *REFINE_ESTIMATES(16)* refinement rule is not required when any of the other rules are enabled.

The following examples show how refinement rules affect estimation on the two ranked indices I22 and I23.

```
SQL> select count(*) from t2 where f2 between 6207 and 6208
cont>    and f3 between 1 and 2;
~S#0003
Tables:
  0 = T2
Aggregate: 0:COUNT (*)
Leaf#01 BgrOnly 0:T2 Card=100000
  Bool: (0.F2 >= 6207) AND (0.F2 <= 6208) AND (0.F3 >= 1) AND (0.F3 <= 2)
  BgrNdx1 I22 [1:1] Fan=17
    Keys: (0.F2 >= 6207) AND (0.F2 <= 6208)
  BgrNdx2 I23 [1:1] Fan=17
    Keys: (0.F3 >= 1) AND (0.F3 <= 2)
~Estim  Ndx1 Ranked: Nodes=2, Min=2, Est=2 Precise IO=4
~Estim  RLEAF Cardinality=  1.0000000E+05
~Estim  Ndx2 Ranked: Nodes=0, Min=0, Est=12250 Descend IO limit IO=4
~E#0003.01(1) Estim   Index/Estimate 1/2 2_12250
```

Because the estimate for the first background index is two rows, the estimation process is limited to two IO's. The second background index does not get estimated because estimation on the first index has already used 4 IO's.

```
SQL> select count(*) from t2 where f2 between 6207 and 6208
cont>    and f3 between 1 and 2;
~S#0004
Tables:
  0 = T2
Aggregate: 0:COUNT (*)
```

```
Leaf#01 BgrOnly 0:T2 Card=100000
  Bool: (0.F2 >= 6207) AND (0.F2 <= 6208) AND (0.F3 >= 1) AND (0.F3 <= 2)
  BgrNdx1 I22 [1:1] Fan=17
    Keys: (0.F2 >= 6207) AND (0.F2 <= 6208)
  BgrNdx2 I23 [1:1] Fan=17
    Keys: (0.F3 >= 1) AND (0.F3 <= 2)
~Estim  Ndx1 Ranked: Nodes=2, Min=2, Est=2 Precise IO=0
~Estim  RLEAF Cardinality=  1.0000000E+05
~Estim  Ndx2 Ranked: Nodes=154, Min=0, Est=3103 Descend IO limit IO=2
~Estim  RLEAF Cardinality=  1.0000000E+05
~E#0004.01(1) Estim   Index/Estimate 1/2 2/3103
```

Because we immediately repeat the same query, the first index is again estimated at two rows, but because the previous query already read these index nodes, this did not cost any IO's because the nodes remained in our buffer pool. This meant that two IO's could be used to begin estimation on the second background index. The estimate is not very accurate because we could only descend a short way into the index structure in two IO's.

In the following example, you will see that we descend further and further down the index on each execution and the estimate becomes progressively better on each execution.

```
SQL> select count(*) from t2 where f2 between 6207 and 6208
cont>    and f3 between 1 and 2;
~S#0005
Tables:
  0 = T2
Aggregate: 0:COUNT (*)
Leaf#01 BgrOnly 0:T2 Card=100000
  Bool: (0.F2 >= 6207) AND (0.F2 <= 6208) AND (0.F3 >= 1) AND (0.F3 <= 2)
  BgrNdx1 I22 [1:1] Fan=17
    Keys: (0.F2 >= 6207) AND (0.F2 <= 6208)
  BgrNdx2 I23 [1:1] Fan=17
    Keys: (0.F3 >= 1) AND (0.F3 <= 2)
~Estim  Ndx1 Ranked: Nodes=2, Min=2, Est=2 Precise IO=0
~Estim  RLEAF Cardinality=  1.0000000E+05
~Estim  Ndx2 Ranked: Nodes=1, Min=0, Est=10 Descend IO limit IO=2
~Estim  RLEAF Cardinality=  1.0000000E+05
~E#0005.01(1) Estim   Index/Estimate 1/2 2/10
~E#0005.01(1) BgrNdx1 EofData  DBKeys=2  Fetches=0+0  RecsOut=0 #Bufs=1
~E#0005.01(1) BgrNdx2 FtchLim  DBKeys=0  Fetches=0+0  RecsOut=0
~E#0005.01(1) Fin     Buf      DBKeys=2  Fetches=0+0  RecsOut=0

           0
1 row selected
SQL> select count(*) from t2 where f2 between 6207 and 6208
cont>    and f3 between 1 and 2;
~S#0006
Tables:
  0 = T2
Aggregate: 0:COUNT (*)
Leaf#01 BgrOnly 0:T2 Card=100000
  Bool: (0.F2 >= 6207) AND (0.F2 <= 6208) AND (0.F3 >= 1) AND (0.F3 <= 2)
  BgrNdx1 I22 [1:1] Fan=17
    Keys: (0.F2 >= 6207) AND (0.F2 <= 6208)
  BgrNdx2 I23 [1:1] Fan=17
    Keys: (0.F3 >= 1) AND (0.F3 <= 2)
~Estim  Ndx1 Ranked: Nodes=2, Min=2, Est=2 Precise IO=0
~Estim  RLEAF Cardinality=  1.0000000E+05
~Estim  Ndx2 Ranked: Nodes=1, Min=2, Est=2 Precise IO=0
```

```
~Estim  RLEAF Cardinality=  1.0000000E+05
~E#0006.01(1) Estim   Index/Estimate 1/2 2/2
```

In this case, after three executions the estimate obtained is precisely two.

It is anticipated that refinement rules will become the default in a future version of Oracle Rdb.

## 7.3.22 Hash Index Estimation

In addition to the functionality described in Section 7.3.21, the index estimation process has been enhanced to support estimation of indexes of *TYPE IS HASHED*.

For a complete description of this feature, please refer to the Rdb Technical Note available through MetaLink.

By default, Rdb does not allow estimation of hashed indexes. As with the new features for *TYPE IS SORTED RANKED*, this feature is controlled using the refine estimates flag.

The values that effect the behaviour on hashed indexes are:

- REFINE_ESTIMATES(32) − Enable estimation on hashed indexes.
- REFINE_ESTIMATES(64) − Use the smallest estimate obtained for this execution of the request to limit the IO consumed during estimation.

The values can be combined with those affecting ranked indices by adding them together.

In the following query, index I32 is *TYPE IS SORTED RANKED* and index I33 is *TYPE IS HASHED*.

```
SQL> set flags 'strategy,detail(1),exec,refine_estimates(111)'
SQL> select count(*) from t2 where f2 between 6207 and 6208
cont> and f3 in (1,2);
~S#0004
Tables:
  0 = T2
Aggregate: 0:COUNT (*)
Leaf#01 BgrOnly 0:T2 Card=100000
  Bool: (0.F2 >= 6207) AND (0.F2 <= 6208) AND ((0.F3 = 1) OR (0.F3 = 2))
  BgrNdx1 I23 [(1:1)2] Fan=1
    Keys: r0: 0.F3 = 2
          r1: 0.F3 = 1
  BgrNdx2 I22 [1:1] Fan=17
    Keys: (0.F2 >= 6207) AND (0.F2 <= 6208)
~Estim  Ndx1 Hashed: Nodes=0, Est=2 Precise IO=5
~Estim  Ndx2 Ranked: Nodes=0, Min=0, Est=12250 Descend IO limit IO=5
~E#0004.01(1) Estim   Index/Estimate 1/2 2_12250
```

The hashed index is background index 1 and is estimated to return precisely two rows. The Refinement rules are in place for ranked indexes so the estimation on the ranked index I22 is not performed.

By repeating the same query, so that some index nodes are already buffered, we can see that estimation on the ranked index will proceed.

Please note that this example has been edited for brevity.

```
SQL> select count(*) from t2 where f2 between 6207 and 6208
cont> and f3 in (1,2);
~Estim  Ndx1 Hashed: Nodes=0, Est=2 Precise IO=0
~Estim  Ndx2 Ranked: Nodes=163, Min=0, Est=3284 Refine IO limit IO=2
~Estim  RLEAF Cardinality=  1.0000000E+05
~E#0005.01(1) Estim   Index/Estimate 1/2 2/3284
~E#0005.01(1) BgrNdx1 EofData  DBKeys=2  Fetches=0+0  RecsOut=0 #Bufs=1
~E#0005.01(1) BgrNdx2 FtchLim  DBKeys=0  Fetches=0+0  RecsOut=0
~E#0005.01(1) Fin     Buf      DBKeys=2  Fetches=0+0  RecsOut=0

           0
1 row selected
SQL> select count(*) from t2 where f2 between 6207 and 6208
cont> and f3 in (1,2);
~Estim  Ndx1 Hashed: Nodes=0, Est=2 Precise IO=0
~Estim  Ndx2 Ranked: Nodes=10, Min=0, Est=190 Refine IO limit IO=2
~Estim  RLEAF Cardinality=  1.0000000E+05
~E#0006.01(1) Estim   Index/Estimate 1/2 2/190
~E#0006.01(1) BgrNdx1 EofData  DBKeys=2  Fetches=0+0  RecsOut=0 #Bufs=1
~E#0006.01(1) BgrNdx2 FtchLim  DBKeys=0  Fetches=0+0  RecsOut=0
~E#0006.01(1) Fin     Buf      DBKeys=2  Fetches=0+0  RecsOut=0

           0
1 row selected
SQL> select count(*) from t2 where f2 between 6207 and 6208
cont> and f3 in (1,2);
~Estim  Ndx1 Hashed: Nodes=0, Est=2 Precise IO=0
~Estim  Ndx2 Ranked: Nodes=2, Min=2, Est=2 Precise IO=0
~Estim  RLEAF Cardinality=  1.0000000E+05
~E#0007.01(1) Estim   Index/Estimate 1/2 2/2
```

Unlike estimation on sorted indexes, estimation of indexes of *TYPE IS HASHED* is performed even where the index has more than one partition.

Estimation on hashed indexes is also supported for range list queries. A range list query, such as the one shown above, provides multiple key values to be retrieved from the same index. This occurs where the query has a condition such as *KEY=42 OR KEY=6* or *KEY IN (1,5,9)*.

## 7.3.23 RMU /VERIFY Enhanced to Detect Sequence Problems

RMU /VERIFY has been enhanced to detect two sequence problems. Each sequence that is created has an CLTSEQ entry and a row in the system table RDB$SEQUENCES corresponding to it.

RMU /VERIFY now makes sure that each row in the RDB$SEQUENCES table has a matching CLTSEQ entry in the root, and each CLTSEQ entry in the root file that is marked "Reserved" (i.e. it is associated with a sequence that is being used) has a matching row in the RDB$SEQUENCES table.

On finding these problems, messages of the following type will be generated:

```
%RMU−E−NOSEQROW, Sequence id 1 has an entry in the root file but no row
in RDB$SEQUENCES
%RMU−E−NOSEQENT, Sequence id 1 has no valid entry in the root file
```

This RMU /VERIFY enhancement is available starting with Oracle Rdb Release 7.1.2.

## 7.3.24 Determining Which Oracle Rdb Options Are Installed

When installing Oracle Rdb Server on OpenVMS you can choose from five components to install:

1. Oracle Rdb
2. Programmer for Rdb (Rdb Compilers)
3. Hot Standby
4. Power Utilities
5. Common Components

Starting with Rdb 7.0, you can determine what Rdb options were selected during the installation of Rdb by running the program SYS$SYSTEM:RDBINS<RdbVersionVariant>.EXE. For example:

```
$ RUN SYS$SYSTEM:RDBINS71
Installed: Oracle Rdb,Rdb Compilers,Hot Standby,Power Utilities
```

Previously, however, the output of the RDBINS program could not easily be redirected. Attempts to redefine SYS$OUTPUT would not allow the program output to be captured.

This problem has been resolved. The RDBINS program now allows redirection of the output to SYS$OUTPUT. The RDBINS program also creates a DCL symbol RDB$INSTALLED_SELECTIONS containing the same output string as is displayed to SYS$OUTPUT.

## 7.3.25 New Procedure RDB$IMAGE_VERSIONS.COM

The command procedure RDB$IMAGE_VERSIONS.COM is supplied in SYS$LIBRARY by the Rdb installation procedure. The RDB$IMAGE_VERSIONS command procedure can be used to display the image identification string and image link date/time from various Oracle Rdb or potentially related images in SYS$SYSTEM, SYS$LIBRARY and SYS$MESSAGE. This procedure can be used to determine exactly what images are installed on the system.

RDB$IMAGE_VERSIONS.COM accepts an optional parameter. If passed, this parameter specifies a specific file or wildcard to lookup and display information for. By default, filenames starting with RD*, SQL*, RM*, and COSI* and ending with .EXE are searched for and displayed.

The following example shows how to use the RDB$IMAGE_VERSIONS command procedure.

```
Decrdb RTA1:> @RDB$IMAGE_VERSIONS
SYS$SYSROOT:[SYSEXE]RDB$NATCONN71.EXE;1 SQL*NET V7.1-55   8-MAY-2002 15:56
SYS$COMMON:[SYSEXE]RDBINS.EXE;4         ORACLE RDB V7.0  14-NOV-2002 17:20
SYS$COMMON:[SYSEXE]RDBINS70.EXE;33      ORACLE RDB V7.0   7-MAR-2003 15:30
SYS$COMMON:[SYSEXE]RDBINS71.EXE;5       ORACLE RDB V7.1   9-APR-2003 10:58
SYS$COMMON:[SYSEXE]RDBPRE.EXE;5         V7.0-65          10-SEP-2002 16:02
SYS$COMMON:[SYSEXE]RDBPRE70.EXE;37      V7.0-7           28-FEB-2003 23:24
SYS$COMMON:[SYSEXE]RDBPRE71.EXE;5       V7.1-101          8-APR-2003 16:49
SYS$COMMON:[SYSEXE]RDBSERVER.EXE;9      RDB/RSV V7.0-65   5-SEP-2002 21:01
SYS$COMMON:[SYSEXE]RDBSERVER70.EXE;41   RDB/RSV V7.0-7   27-FEB-2003 17:29
SYS$COMMON:[SYSEXE]RDBSERVER71.EXE;5    RDB/RSVV7.1-101   7-APR-2003 17:43
SYS$COMMON:[SYSEXE]RDMABS.EXE;5         RDB V7.0-65      10-SEP-2002 16:01
     .
     .
     .
```

### 7.3.26 Oracle Rdb SGA API

Oracle Rdb maintains an extensive set of online performance statistics that provide valuable dynamic information regarding the status of an active database. The system global area (SGA) application programming interface (API) described in this document provides a way to retrieve these database performance statistics.

The SGA API automates retrieving database statistics available only through the RMU Show Statistics command. The SGA API provides the only way to retrieve statistics for Oracle Rdb databases from an application. Using the SGA API provides fast access to the data without affecting the execution of the server.

Previously, the Oracle Rdb SGA API was available as a separate software option to be downloaded, installed and maintained independently of the Oracle Rdb kit. Each time a new version of Oracle Rdb was installed, the SGA API would have to be updated. If the SGA API was not updated, it would in many cases fail to work correctly.

This problem has been partly resolved. Most of the contents of the SGA API separate software option are now automatically provided in the RDM$DEMO directory during the Oracle Rdb kit installation procedure. Please refer to the SGA API documentation available in RDM$DEMO in various formats as SGAAPI.PS, SGAAPI.HTML and SGAAPI.TXT for additional information.

Existing Users of the SGAAPI May Have to Modify Procedures

***Existing users of the Oracle Rdb SGA API should refer to the documentation as there will be some minor changes required. In particular, the KUSRMUSHRxx.EXE sharable image is now provided in SYS$LIBRARY and the KUSRMUSHRxx.OPT linker options file has been updated to reference this sharable image in its new location.***

# Chapter 8
# Documentation Corrections, Additions and Changes

This chapter provides corrections for documentation errors and omissions.

# 8.1 Documentation Corrections

## 8.1.1 Database Server Process Priority Clarification

By default, the database servers (ABS, ALS, DBR, LCS, LRS, RCS) created by the Rdb monitor inherit their VMS process scheduling base priority from the Rdb monitor process. The default priority for the Rdb monitor process is 15.

Individual server priorities can be explicitly controlled via system−wide logical names as described in Table 8−1.

***Table 8−1 Server Process Priority Logical Names***

| Logical Name | Use |
|---|---|
| RDM$BIND_ABS_PRIORITY | Base Priority for the ABS Server process |
| RDM$BIND_ALS_PRIORITY | Base Priority for the ALS Server process |
| RDM$BIND_DBR_PRIORITY | Base Priority for the DBR Server process |
| RDM$BIND_LCS_PRIORITY | Base Priority for the LCS Server process |
| RDM$BIND_LRS_PRIORITY | Base Priority for the LRS Server process |
| RDM$BIND_RCS_PRIORITY | Base Priority for the RCS Server process |

When the Hot Standby feature is installed, the RDMAIJSERVER account is created specifying an account priority of 15. The priority of AIJ server processes on your system can be restricted with the system−wide logical name RDM$BIND_AIJSRV_PRIORITY. If this logical name is defined to a value less than 15, an AIJ server process will adjust its base priority to the value specified when the AIJ server process starts. Values from 0 to 31 are allowed for RDM$BIND_AIJSRV_PRIORITY, but the process is not able to raise its priority above the RDMAIJSERVER account value.

For most applications and systems, Oracle discourages changing the server process priorities.

## 8.1.2 Explanation of SQL$INT in a SQL Multiversion Environment and How to Redefine SQL$INT

Bug 2500594

In an environment running multiple versions of Oracle Rdb, for instance Rdb V7.0 and Rdb V7.1, there are now several varianted SQL images, such as SQL$70.EXE and SQL$71.EXE. However, SQL$INT.EXE is not varianted but acts as a dispatcher using the translation of the logical name RDMS$VERSION_VARIANT to activate the correct SQL runtime environment. This image is replaced when a higher version of Oracle Rdb is installed. Thus, using the example above, when Rdb V7.1 is installed, SQL$INT.EXE will be replaced with the V7.1 SQL$INT.EXE.

If an application is linked in this environment (using V7.1 SQL$INT) and the corresponding executable deployed to a system running Oracle Rdb V7.0 multiversion only, the execution of the application may result in the following error:

```
%IMGACT−F−SYMVECMIS, shareable image's symbol vector table mismatch
```

In order to avoid such a problem, the following alternative is suggested:

In the multiversion environment running both Oracle Rdb V7.0 and Oracle Rdb V7.1, run Oracle Rdb V7.0 multiversion by running the command procedures RDB$SETVER.COM 70 and RDB$SETVER RESET. This will set up the necessary logical names and symbols that establish the Oracle Rdb V7.0 environment.

For example:

```
$ @SYS$LIBRARY:RDB$SETVER 70

Current PROCESS Oracle Rdb environment is version V7.0−63 (MULTIVERSION)
Current PROCESS SQL environment is version V7.0−63 (MULTIVERSION)
Current PROCESS Rdb/Dispatch environment is version V7.0−63 (MULTIVERSION)

$ @SYS$LIBRARY:RDB$SERVER RESET
```

Now run SQL and verify that the version is correct:

```
$ sql$
SQL> show version
Current version of SQL is: Oracle Rdb SQL V7.0−63
```

Define SQL$INT to point to the varianted SQL$SHR.EXE. Then, create an options file directing the linker to link with this newly defined SQL$INT. An example follows:

```
$ DEFINE SQL$INT SYS$SHARE:SQL$SHR'RDMS$VERSION_VARIANT'.EXE
$ LINK TEST_APPL,SQL$USER/LIB,SYS$INPUT/option
SQL$INT/SHARE
^Z
```

The executable is now ready to be deployed to the Oracle Rdb V7.0 multiversion environment and should run successfully.

Please note that with each release of Oracle Rdb, new entry points are added to the SQL$INT shareable image. This allows the implementation of new functionality. Therefore, applications linked with SQL$INT from Oracle Rdb V7.1 cannot be run on systems with only Oracle Rdb V7.0 installed. This is because the shareable image does not contain sufficient entry points.

The workaround presented here allows an application to explicitly link with the Oracle Rdb V7.0 version of the image. Such applications are upward compatible and will run on Oracle Rdb V7.0 and Oracle Rdb V7.1. The applications should be compiled and linked under the lowest version.

In environments where Oracle Rdb V7.1 is installed, this workaround is not required because the SQL$INT image will dynamically activate the appropriate SQL$SHRxx image as expected.

### 8.1.3 Documentation Omitted Several Reserved Words

Bug 2319321

The following keywords are considered reserved words in Oracle Rdb Release 7.1.

- UID
- CURRENT_UID
- SYSTEM_UID
- SESSION_UID
- RAW
- LONG
- DBKEY
- ROWID
- SYSDATE

In particular, any column which has these names will be occluded by the keyword. i.e. selecting from column UID will be interpreted as referencing the built in function UID and so return a different result.

The correction to this problem is to enable keyword quoting using SET QUOTING RULES 'SQL92' (or 'SQL99') and enclose the column name in quotations.

In addition, SQL will now generate a warning if these reserved words are used (unquoted) in CREATE and ALTER operations.

### 8.1.4 Using Databases from Releases Earlier Than V6.0

Bug 2383967

You cannot convert or restore databases earlier than V6.0 directly to V7.1. The RMU Convert command for V7.1 supports conversions from V6.0 through V7.0 only. If you have a V3.0 through V5.1 database, you must convert it to at least V6.0 and then convert it to V7.1. For example, if you have a V4.2 database, convert it first to at least V6.0, then convert the resulting database to V7.1.

If you attempt to convert a database created prior to V6.0 directly to V7.1, Oracle RMU generates an error.

### 8.1.5 Clarification of PREPARE Statement Behavior

Bug 2581863

According to the Oracle Rdb7 SQL Reference Manual, Volume 3 page 7−227, when using a statement−id parameter for PREPARE "if that parameter is an integer, then you must explicitly initialize that integer to zero before executing the PREPARE statement".

This description is not correct and should be replaced with this information:

1. If the statement−id is non−zero and does not match any prepared statement (the id was stale or contained a random value), then an error is raised:

%SQL−F−BADPREPARE, Cannot use DESCRIBE or EXECUTE on a statement that is not prepared

2. If the statement−id is non−zero, or the statement name is one that has previously been used and matches an existing prepared statement, then that statement is automatically released prior to the prepare of the new statement. Please refer to the RELEASE statement for further details.
3. If the statement−id is zero or was automatically released, then a new statement−id is allocated and the statement prepared.

Please note that if you use statement−name instead of a statement−id−parameter then SQL will implicitly declare an id for use by the application. Therefore, the semantics described apply similarly when using the statement−name. See the RELEASE statement for details.

## 8.1.6 New RMU/BACKUP Storage Area Assignment With Thread Pools

This is to clarify how storage areas are assigned to disk and tape devices using the new RMU/BACKUP THREAD POOL and BACKUP TO MULTIPLE DISK DEVICES features introduced in Oracle Rdb Release 7.1.

For the case of backup to multiple disk devices using thread pools, the algorithm used by RMU/BACKUP to assign threads is to calculate the size of each area as the product of the page length in bytes times the highest page number used (maximum page number) for that area. The area sizes are then sorted by descending size and ascending device name. For internal processing reasons, the system area is placed as the first area in the first thread. Each of the remaining areas is added to whichever thread has the lowest byte count. In this way, the calculated area sizes are balanced between the threads.

For tape devices, the same algorithm is used but the areas are partitioned among writer threads, not disk devices.

The partitioning for backup to multiple disk devices is done by disk device, not by output thread, because there will typically be more disk devices than output threads, and an area can not span a device.

## 8.1.7 RDM$BIND_LOCK_TIMEOUT_INTERVAL Overrides the Database Parameter

Bug 2203700

When starting a transaction, there are three different values that are used to determine the lock timeout interval for that transaction. Those values are:

1. The value specified in the SET TRANSACTION statement
2. The value stored in the database as specified in CREATE or ALTER DATABASE
3. The value of the logical name RDM$BIND_LOCK_TIMEOUT_INTERVAL

The timeout interval for a transaction is the smaller of the value specified in the SET TRANSACTION statement and the value specified in CREATE DATABASE. However, if the logical name RDM$BIND_LOCK_TIMEOUT_INTERVAL is defined, the value of this logical name overrides the value specified in CREATE DATABASE.

The description of how these three values interact, found in several different parts of the Rdb documentation set, is incorrect and will be replaced by the description above.

The lock timeout value in the database can be dynamically modified from the Locking Dashboard in RMU/SHOW STATISTICS. The Per−Process Locking Dashboard can be used to dynamically override the logical name RDM$BIND_LOCK_TIMEOUT_INTERVAL for one or more processes.

## 8.1.8 New Request Options for RDO, RDBPRE and RDB$INTERPRET

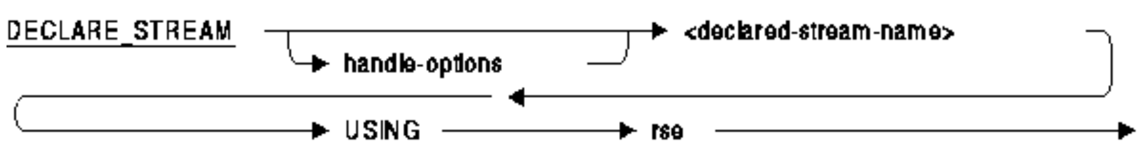
This release note was included in the V70A Release Notes but had gotten dropped somewhere along the line.

For this release of Rdb, two new keywords have been added to the handle−options for the DECLARE_STREAM, the START_STREAM (undeclared format) and FOR loop statements. These changes have been made to RDBPRE, RDO and RDB$INTERPRET at the request of several RDO customers.

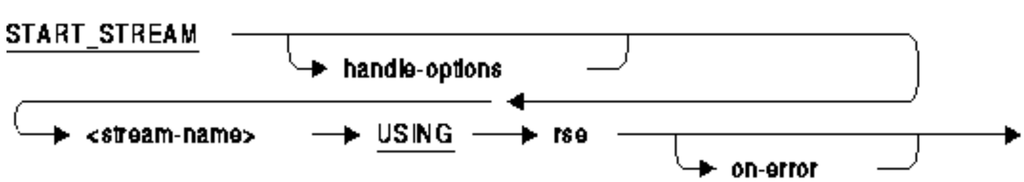
In prior releases, the handle−options could not be specified in interactive RDO or RDB$INTERPRET. This has changed in Rdb7 but these allowed options will be limited to MODIFY and PROTECTED keywords. For RDBPRE, all options listed will be supported. These option names were chosen to be existing keywords to avoid adding any new keywords to the RDO language.

The altered statements are shown in Example 5−1, Example 5−2 and Example 5−3.

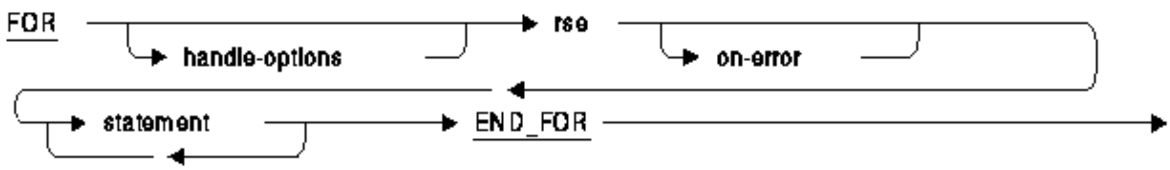
Example 5−1 DECLARE_STREAM Format



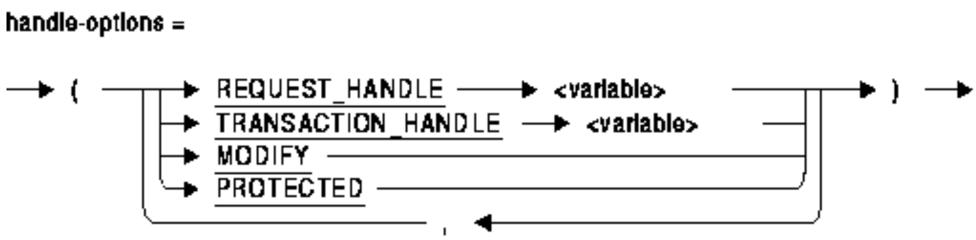

Example 5−2 START_STREAM Format




Example 5−3 FOR Format




Each of these statements references the syntax for the HANDLE−OPTIONS which has been revised and is shown below.




The following options are available for HANDLE−OPTIONS:

- REQUEST_HANDLE specifies the request handle for this request. This option is only valid for RDBPRE and RDML applications. It cannot be used with RDB$INTERPRET, nor interactive RDO.
- TRANSACTION_HANDLE specifies the transaction handle under which this request executes. This option is only valid for RDBPRE and RDML applications. It cannot be used with RDB$INTERPRET, nor interactive RDO.
- MODIFY specifies that the application will modify all (or most) records fetched from the stream or for loop. This option can be used to improve application performance by avoiding lock promotion from SHARED READ for the FETCH to PROTECTED WRITE access for the nested MODIFY or ERASE statement. It can also reduce DEADLOCK occurrence because lock promotions are avoided.
  This option is valid for RDBPRE, RDB$INTERPRET, and interactive RDO. This option is not currently available for RDML.
  For example:

  ```
  RDO>   FOR (MODIFY) E IN EMPLOYEES WITH E.EMPLOYEE_ID = "00164"
  cont>    MODIFY E USING E.MIDDLE_INITIAL = "M"
  cont>    END_MODIFY
  cont>  END_FOR
  ```

  This FOR loop uses the MODIFY option to indicate that the nested MODIFY is an unconditional statement and so aggressive locking can be undertaken during the fetch of the record in the FOR loop.
- PROTECTED specifies that the application may modify records fetched by this stream by a separate and independent MODIFY statement. Therefore, this stream should be protected from interference (aka Halloween affect). The optimizer will select a snapshot of the rows and store them in a temporary relation for processing, rather than traversing indexes at the

time of the FETCH statement. In some cases this may result in poorer performance when the temporary relation is large and overflows from virtual memory to a temporary disk file, but the record stream will be protected from interference. The programmer is directed to the documentation for the Oracle Rdb logical names RDMS$BIND_WORK_VM and RDMS$BIND_WORK_FILE.

This option is valid for RDBPRE, RDB$INTERPRET, and interactive RDO. This option is not currently available for RDML.

The following example creates a record stream in a BASIC program using Callable RDO:

```
RDMS_STATUS = RDB$INTERPRET ('INVOKE DATABASE PATHNAME "PERSONNEL"')

RDMS_STATUS = RDB$INTERPRET ('START_STREAM (PROTECTED) EMP USING ' + &
                             'E IN EMPLOYEES')

RDMS_STATUS = RDB$INTERPRET ('FETCH EMP')

DML_STRING = 'GET ' +                                           &
                 '!VAL = E.EMPLOYEE_ID;' +                      &
                 '!VAL = E.LAST_NAME;' +                        &
                 '!VAL = E.FIRST_NAME' +                        &
             'END_GET'

RDMS_STATUS = RDB$INTERPRET (DML_STRING, EMP_ID, LAST_NAME, FIRST_NAME)
```

In this case the FETCH needs to be protected against MODIFY statements which execute in other parts of the application.

# 8.2 Address and Phone Number Correction for Documentation

In release 7.0 or earlier documentation, the address and fax phone number listed on the Send Us Your Comments page are incorrect. The correct information is:

```
FAX -- 603.897.3825
Oracle Corporation
One Oracle Drive
Nashua, NH 03062-2804
USA
```

## 8.3 Online Document Format and Ordering Information

You can view the documentation in Adobe Acrobat format using the Acrobat Reader, which allows anyone to view, navigate, and print documents in the Adobe Portable Document Format (PDF). See http://www.adobe.com for information about obtaining a free copy of Acrobat Reader and for information on supported platforms.

The Oracle Rdb documentation in Adobe Acrobat format is available on MetaLink:

```
Top Tech Docs\Oracle Rdb\Documentation\<bookname>
```

Customers should contact their Oracle representative to purchase printed documentation.

# 8.4 New and Changed Features in Oracle Rdb Release 7.1

This section provides information about late−breaking new features or information that is missing or changed since the Oracle Rdb New and Changed Features for Oracle Rdb manual was published.

## 8.4.1 PERSONA is Supported in Oracle SQL/Services

In the "New and Changed Features for Oracle Rdb" Manual under the section "ALTER DATABASE Statement" is a note stating that impersonation is not supported in Oracle SQL/Services. This is incorrect. There was a problem in the first release of Oracle Rdb 7.1 (7.1.0) whereby impersonation through Oracle SQL/Services failed. This problem is resolved in Oracle Rdb Release 7.1.0.1.

## 8.4.2 NEXTVAL and CURRVAL Pseudocolumns Can Be Delimited Identifiers

The New and Changed Features for Oracle Rdb manual describes SEQUENCES but does not mention that the special pseudocolumns NEXTVAL and CURRVAL can be delimited. All uppercase and lowercase variations of these keywords are accepted and assumed to be equivalent to these uppercase keywords.

The following example shows that any case is accepted:

```
SQL> set dialect 'sql92';
SQL> create sequence dept_id;
SQL> select dept_id.nextval from rdb$database;
                    1
1 row selected
SQL> select "DEPT_ID".currval from rdb$database;
                    1
1 row selected
SQL> select "DEPT_ID"."CURRVAL" from rdb$database;
                    1
1 row selected
SQL> select "DEPT_ID"."nextval" from rdb$database;
                    2
1 row selected
SQL> select "DEPT_ID"."CuRrVaL" from rdb$database;
                    2
1 row selected
```

## 8.4.3 Only=select_list Qualifier for the RMU Dump After_Journal Command

The Oracle Rdb New and Changed Features for Oracle Rdb manual documents the First=select_list and Last=select_list qualifiers for the RMU Dump After_Journal command. Inadvertently missed was the Only=select_list qualifier.

The First, Last, and Only qualifiers have been added because the Start and End qualifiers are difficult to use since users seldom know, nor can they determine, the AIJ record number in advance of using the RMU Dump After_Journal command.

The select_list clause of these qualifiers consists of a list of one or more of the following keywords:

- TSN=tsn
  Specifies the first, last, or specific TSN in the AIJ journal using the standard [n:]m TSN format.
- TID=tid
  Specifies the first, last or specific TID in the AIJ journal.
- RECORD=record
  Specifies the first or last record in the AIJ journal. This is the same as the existing Start and End qualifiers (which are still supported, but deprecated). This keyword cannot be used with the Only qualifier.
- BLOCK=block#
  Specifies the first or last block in the AIJ journal. This keyword cannot be used with the Only qualifier.
- TIME=date_time
  Specifies the first or last date/time in the AIJ journal using the standard date/time format. This keyword cannot be used with the Only qualifier.

The First, Last, and Only qualifiers are optional. You may specify any or none of them.

The keywords specified for the First qualifier can differ from the keywords specified for the other qualifiers.

For example, to start the dump from the fifth block of the AIJ journal, you would use the following command:

```
RMU/DUMP/AFTER_JOURNAL /FIRST=(BLOCK=5) MF_PERSONNEL.AIJ
```

To start the dump from block 100 or TSN 52, whichever occurs first, you would use the following command:

```
RMU/DUMP/AFTER_JOURNAL /FIRST=(BLOCK=100,TSN=0:52) MF_PERSONNEL.AIJ
```

When multiple keywords are specified for a qualifier, the first condition being encountered activates the qualifier. In the preceding example, the dump starts when either block 100 or TSN 52 is encountered.

Be careful when searching for TSNs or TIDs as they are not ordered in the AIJ journal. For example, if you want to search for a specific TSN, use the Only qualifier and not the First and Last qualifiers. For example, assume the AIJ journal contains records for TSN 150, 170, and 160 (in that order). If you specify the First=TSN=160 and Last=TSN=160 qualifiers, nothing will be dumped because TSN 170 will match the Last=TSN=160 criteria.

# 8.5 Oracle Rdb7 and Oracle CODASYL DBMS Guide to Hot Standby Databases

This section provides information that is missing from or changed in V7.0 of the Oracle Rdb7 and Oracle CODASYL DBMS Guide to Hot Standby Databases.

## 8.5.1 Restrictions Lifted on After−Image Journal Files

The Hot Standby software has been enhanced regarding how it handles after−image journal files. Section 4.2.4 in the Oracle Rdb and Oracle CODASYL DBMS Guide to Hot Standby Databases states the following information:

```
If an after−image journal switchover operation is suspended when
replication operations are occurring, you must back up one or more of
the modified after−image journals to add a new journal file.
```

This restriction has been removed. Now, you can add journal files or use the emergency AIJ feature of Oracle Rdb release 7.0 to automatically add a new journal file. Note the following distinctions between adding an AIJ file and adding an emergency AIJ file:

- You can add an AIJ file to the master database and it will be replicated on the standby database. If replication operations are active, the AIJ file is created on the standby database immediately. If replication operations are not active, the AIJ file is created on the standby database when replication operations are restarted.
- You can add emergency AIJ files anytime. If replication operations are active, the emergency AIJ file is created on the standby database immediately. However, because emergency AIJ files are not journaled, starting replication after you create an emergency AIJ will fail. You cannot start replication operations because the Hot Standby software detects a mismatch in the number of after−image journal files on the master compared to the standby database.
  If an emergency AIJ file is created on the master database when replication operations are not active, you must perform a master database backup and then restore the backup on the standby database. Otherwise, an AIJSIGNATURE error results.

## 8.5.2 Changes to RMU Replicate After_Journal ... Buffer Command

The behavior of the RMU Replicate After_Journal ... Buffers command has been changed. The Buffers qualifier may be used with either the Configure option or the Start option.

When using local buffers, the AIJ Log Roll−forward Server will use a minimum of 4096 buffers. The value provided to the Buffers qualifier will be accepted but ignored if it is less than 4096. In addition, further parameters will be checked and the number of buffers may be increased if the resulting calculations are greater than the number of buffers specified by the Buffers qualifier. If the database is configured to use more than 4096 AIJ Request Blocks (ARBs), then the number of buffers may be increased to the number of ARBs configured for the database. The LRS ensures that there are at least 10 buffers for every possible storage area in the database. Thus if the total number of storage areas (both used and reserved) multiplied by 10 results in a greater number of buffers, then that number will be used.

When global buffers are used, the number of buffers used by the AIJ Log Roll−forward Server is determined as follows:

- If the Buffers qualifier is omitted and the Online qualifier is specified, then the number of buffers will default to the previously configured value, if any, or 256, whichever is larger.
- If the Buffers qualifier is omitted and the Online qualifier is not specified or the Noonline qualifier is specified, then the number of buffers will default to the maximum number of global buffers allowed per user ("USER LIMIT"), or 256, whichever is larger.
- If the Buffers qualifier is specified then that value must be at least 256, and it may not be greater than the maximum number of global buffers allowed per user ("USER LIMIT").

The Buffer qualifier now enforces a minimum of 256 buffers for the AIJ Log Roll−forward Server. The maximum number of buffers allowed is still 524288 buffers.

## 8.5.3 Unnecessary Command in the Hot Standby Documentation

There is an unnecessary command documented in the Oracle Rdb and Oracle CODASYL DBMS Guide to Hot Standby Databases manual. The documentation (in Section 2.12 "Step 10: Specify the Network Transport Protocol") says that to use TCP/IP as the network protocol, you must issue the following commands:

```
$ CONFIG UCX AIJSERVER OBJECT
$ UCX SET SERVICE RDMAIJSRV
/PORT=n
/USER_NAME=RDMAIJSERVER
/PROCESS_NAME=RDMAIJSERVER
/FILE=SYS$SYSTEM:rdmaijserver_ucx.com
/LIMIT=nn
```

The first of these commands ($ CONFIG UCX AIJSERVER OBJECT) is unnecessary. You can safely disregard the first line when setting up to use TCP/IP with Hot Standby.

The documentation will be corrected in a future release of Oracle Rdb.

## 8.5.4 Change in the Way RDMAIJ Server is Set Up in UCX

Starting with Oracle Rdb Release 7.0.2.1, the RDMAIJ image became a varianted image. Therefore, the information in Section 2.12, "Step 10: Specify the Network Transport Protocol," of the Oracle Rdb7 and Oracle CODASYL DBMS Guide to Hot Standby Databases has become outdated with regard to setting up the RDMAIJSERVER object when using UCX as the network transport protocol. The UCX SET SERVICE command is now similar to the following:

```
$ UCX SET SERVICE RDMAIJ −
      /PORT=port_number −
      /USER_NAME=RDMAIJ −
      /PROCESS_NAME=RDMAIJ −
      /FILE=SYS$SYSTEM:RDMAIJSERVER.com −
      /LIMIT=limit
```

For Oracle Rdb multiversion, the UCX SET SERVICE command is similar to the following:

```
$ UCX SET SERVICE RDMAIJ70 −
```

```
/PORT=port_number -
/USER_NAME=RDMAIJ70 -
/PROCESS_NAME=RDMAIJ70 -
/FILE=SYS$SYSTEM:RDMAIJSERVER70.com -
/LIMIT=limit
```

The installation procedure for Oracle Rdb creates a user named RDMAIJ(nn) and places a file called RDMAIJSERVER(nn).COM in SYS$SYSTEM. The RMONSTART(nn).COM command procedure will try to enable a service called RDMAIJ(nn) if UCX is installed and running.

Changing the RDMAIJ server to a varianted image does not impact installations using DECNet since the correct DECNet object is created during the Oracle Rdb installation.

## 8.5.5 CREATE INDEX Operation Supported for Hot Standby

On Page 1−13 of the Oracle Rdb7 and Oracle CODASYL DBMS Guide to Hot Standby Databases, the add new index operation is incorrectly listed as an offline operation not supported by Hot Standby. The CREATE INDEX operation is now fully supported by Hot Standby, as long as the transaction does not span all available AIJ journals, including emergency AIJ journals.

# 8.6 Oracle Rdb7 for OpenVMS Installation and Configuration Guide

This section provides information that is missing from or changed in V7.0 of the Oracle Rdb7 for OpenVMS Installation and Configuration Guide.

## 8.6.1 Suggestion to Increase GH_RSRVPGCNT Removed

The Oracle Rdb7 for OpenVMS Installation and Configuration Guide contains a section titled "Installing Oracle Rdb Images as Resident on OpenVMS Alpha". This section includes information about increasing the value of the OpenVMS system parameter GH_RSRVPGCNT when you modify the RMONSTART.COM or SQL$STARTUP.COM procedures to install Oracle Rdb images with the Resident qualifier.

Note that modifying the parameter GH_RSRVPGCNT is only required if the RMONSTART.COM or SQL$STARTUP.COM procedures have been manually modified to install Oracle Rdb images with the Resident qualifier. Furthermore, if the RMONSTART.COM and SQL$STARTUP.COM procedures are executed during the system startup procedure (directly from SYSTARTUP_VMS.COM, for example), there is no need to modify the GH_RSRVPGCNT parameter.

Oracle Corporation recommends that you do not modify the value of the GH_RSRVPGCNT system parameter unless it is absolutely required. Some versions of OpenVMS on some hardware platforms require GH_RSRVPGCNT to be a value of zero in order to ensure the highest level of system performance.

## 8.6.2 Prerequisite Software

In addition to the software listed in the Oracle Rdb Installation and Configuration Guide and at the url http://www.oracle.com/rdb/product_info/index.html, note that the MACRO−32 compiler and the OpenVMS linker are required OpenVMS components in order to install Oracle Rdb on your OpenVMS Alpha system.

The MACRO−32 Compiler for OpenVMS Alpha is a standard component of the OpenVMS Operating System. It is used to compile :VAX MACRO assembly language source files into native OpenVMS Alpha object code. During the Oracle Rdb installation procedure, and portions of the installation verification procedure (such as the test for RDBPRE), the MACRO−32 compiler is required.

The OpenVMS linker is a standard component of the OpenVMS Operating System. It is used to link one or more input files into a program image and defines the execution characteristics of the image. The linker will be required for application development and is likewise used by the Oracle Rdb installation procedure and the installation verification procedure.

## 8.6.3 Defining the RDBSERVER Logical Name

Sections 4.3.7.1 and 4.3.7.2 in the Oracle Rdb7 for OpenVMS Installation and Configuration Guide provide the following examples for defining the RDBSERVER logical name: *$ DEFINE*

*RDBSERVER SYS$SYSTEM:RDBSERVER70.EXE*

and *$ DEFINE RDBSERVER SYS$SYSTEM:RDBSERVER61.EXE*

These definitions are inconsistent with other command procedures that attempt to reference the RDBSERVERxx.EXE image. Below is one example where the RDBSERVER.COM procedure references SYS$COMMON:<SYSEXE> and SYS$COMMON:[SYSEXE] rather than SYS$SYSTEM.

```
$   if .not. -
       ((f$locate ("SYS$COMMON:<SYSEXE>",rdbserver_image) .ne. log_len) .or. -
        (f$locate ("SYS$COMMON:[SYSEXE]",rdbserver_image) .ne. log_len))
$   then
$       say "''rdbserver_image' is not found in SYS$COMMON:<SYSEXE>"
$       say "RDBSERVER logical is ''rdbserver_image'"
$       exit
$   endif
```

In this case, if the logical name were defined as instructed in the Oracle Rdb7 for OpenVMS Installation and Configuration Guide, the image would not be found.

The correct definition of the logical name is as follows: *DEFINE RDBSERVER SYS$COMMON:<SYSEXE>RDBSERVER70.EXE*

and *DEFINE RDBSERVER SYS$COMMON:<SYSEXE>RDBSERVER61.EXE*

# 8.7 Guide to Database Design and Definition

This section provides information that is missing from or changed in release 7.0 of the Oracle Rdb7 Guide to Database Design and Definition.

## 8.7.1 Lock Timeout Interval Logical Incorrect

On Page 7−31 of Section 7.4.8 in the Oracle Rdb7 Guide to Database Design and Definition, the RDM$BIND_LOCK_TIMEOUT logical name is referenced incorrectly. The correct logical name is RDM$BIND_LOCK_TIMEOUT_INTERVAL.

The Oracle Rdb7 Guide to Database Design and Definition will be corrected in a future release.

## 8.7.2 Example 4−13 and Example 4−14 Are Incorrect

Example 4−13 showing vertical partitioning, and Example 4−14, showing vertical and horizontal partitioning, are incorrect. They should appear as follows:

Example 4−13:

```
SQL> CREATE STORAGE MAP EMPLOYEES_1_MAP
cont>    FOR EMPLOYEES
cont>    ENABLE COMPRESSION
cont>    STORE COLUMNS (EMPLOYEE_ID, LAST_NAME, FIRST_NAME,
cont>                   MIDDLE_INITIAL, STATUS_CODE)
cont>                   DISABLE COMPRESSION
cont>                   IN ACTIVE_AREA
cont>    STORE COLUMNS (ADDRESS_DATA_1, ADDRESS_DATA_2, CITY,
cont>                   STATE, POSTAL_CODE)
cont>                   IN INACTIVE_AREA
cont>    STORE IN OTHER_AREA;
```

Example 4−14:

```
SQL>  CREATE STORAGE MAP EMPLOYEES_1_MAP2
cont>        FOR EMP2
cont>        STORE COLUMNS (EMPLOYEE_ID, LAST_NAME, FIRST_NAME,
cont>                       MIDDLE_INITIAL, STATUS_CODE)
cont>              USING (EMPLOYEE_ID)
cont>              IN ACTIVE_AREA_A WITH LIMIT OF ('00399')
cont>              IN ACTIVE_AREA_B WITH LIMIT OF ('00699')
cont>              OTHERWISE IN ACTIVE_AREA_C
cont>        STORE COLUMNS (ADDRESS_DATA_1, ADDRESS_DATA_2, CITY,
cont>                      STATE, POSTAL_CODE)
cont>              USING (EMPLOYEE_ID)
cont>              IN INACTIVE_AREA_A WITH LIMIT OF ('00399')
cont>              IN INACTIVE_AREA_B WITH LIMIT OF ('00699')
cont>              OTHERWISE IN INACTIVE_AREA_C
cont>        STORE IN OTHER_AREA;
```

# 8.8 Oracle Rdb7 SQL Reference Manual

This section provides information that is missing from or changed in V7.0 of the Oracle Rdb7 SQL Reference Manual.

## 8.8.1 Node Specification Allowed on Root FILENAME Clauses

In previous releases of the Oracle Rdb SQL Reference Manual, it was not made clear that a node specification may only be specified for the root FILENAME clause of the ALTER DATABASE, CREATE DATABASE, EXPORT DATABASE, and IMPORT DATABASE statements.

This means that the directory or file specification specified with the following clauses can only be a device, directory, file name, and file type:

- LOCATION clause of the ROW CACHE IS ENABLED, RECOVERY JOURNAL, ADD CACHE, and CREATE CACHE clauses
- SNAPSHOT FILENAME clause
- FILENAME and SNAPSHOT FILENAME clauses of the ADD STORAGE AREA and CREATE STORAGE AREA clauses
- BACKUP FILENAME clause of the JOURNAL IS ENABLED, ADD JOURNAL, and ALTER JOURNAL clauses
- BACKUP SERVER and CACHE FILENAME clauses of the JOURNAL IS ENABLED clause
- FILENAME clause of the ADD JOURNAL clause

Usage notes reflecting this restriction for these clauses will appear in a future release of the Oracle Rdb SQL Reference Manual.

## 8.8.2 Use of SQL_SQLCA Include File Intended for Host Language File

Use of the SQLCA include files such as the SQL_SQLCA.H file for C, are intended for use with the host language files only. That is, only *.C should be including that file. Precompiled files (*.SC files) should use the EXEC SQL INCLUDE SQLCA embedded SQL command in the declaration section of the module. In this way the precompiler can properly define the structure to be used by the related SQL generated code.

Remember that the SQLCA is always scoped at the module level, unlike the SQLCODE or SQLSTATE variables which may be routine specific.

The following example shows this error:

```
#include <stdio.h>
#include <sql_sqlca.h>
struct SQLCA SQLCA;

int main (void)
{
EXEC SQL EXECUTE IMMEDIATE `show version';
printf ("SQLCODE=%d\n", SQLCA.SQLCODE);
```

```
}
$ SQLPRE/CC issues the following error against this program:
%SQL−F−NOSQLCODE, Neither SQLCA, SQLCODE nor SQLSTATE were declared
```

The following example shows correct usage:

```
#include <stdio.h>
#include <sql_sqlca.h>
EXEC SQL INCLUDE SQLCA;

int main (void)
{
EXEC SQL EXECUTE IMMEDIATE `show version';
printf ("SQLCODE=%d\n", SQLCA.SQLCODE);
}
```

# 8.9 Oracle RMU Reference Manual, Release 7.0

This section provides information that is missing from or changed in V7.0 of the Oracle RMU Reference Manual.

## 8.9.1 RMU Unload After_Journal Null Bit Vector Clarification

Each output record from the RMU /UNLOAD /AFTER_JOURNAL command includes a vector (array) of bits. There is one bit for each field in the data record. If a null bit value is 1, the corresponding field is NULL; if a null bit value is 0, the corresponding field is not NULL and contains an actual data value. The contents of a data field that is NULL are not initialized and are not predictable.

The null bit vector begins on a byte boundary. The field RDB$LM_NBV_LEN indicates the number of valid bits (and thus, the number of columns in the table). Any extra bits in the final byte of the vector after the final null bit are unused and the contents are unpredictable.

The following example C program demonstrates one possible way of reading and parsing a binary output file (including the null bit vector) from the RMU /UNLOAD /AFTER_JOURNAL command. This sample program has been tested using Oracle Rdb V7.0.5 and higher and HP C V6.2−009 on OpenVMS Alpha V7.2−1. It is meant to be used as a template for writing your own program.

```
/* DATATYPES.C */

#include <stdio.h>
#include <descrip.h>
#include <starlet.h>
#include <string.h>

#pragma member_alignment __save
#pragma nomember_alignment

struct { /* Database key structure */
    unsigned short      lno;    /* line number */
    unsigned int        pno;    /* page number */
    unsigned short      dbid;   /* area number */
    } dbkey;

typedef struct { /* Null bit vector with one bit for each column */
    unsigned            n_tinyint   :1;
    unsigned            n_smallint  :1;
    unsigned            n_integer   :1;
    unsigned            n_bigint    :1;
    unsigned            n_double    :1;
    unsigned            n_real      :1;
    unsigned            n_fixstr    :1;
    unsigned            n_varstr    :1;
    } nbv_t;

struct { /* LogMiner output record structure for table DATATYPES */
    char                rdb$lm_action;
    char                rdb$lm_relation_name [31];
    int                 rdb$lm_record_type;
    short               rdb$lm_data_len;
    short               rdb$lm_nbv_len;
```

```
    __int64             rdb$lm_dbk;
    __int64             rdb$lm_start_tad;
    __int64             rdb$lm_commit_tad;
    __int64             rdb$lm_tsn;
    short               rdb$lm_record_version;
    char                f_tinyint;
    short               f_smallint;
    int                 f_integer;
    __int64             f_bigint;
    double              f_double;
    float               f_real;
    char                f_fixstr[10];
    short               f_varstr_len;   /* length of varchar */
    char                f_varstr[10];   /* data of varchar */
    nbv_t               nbv;
    } lm;

#pragma member_alignment __restore

main ()
{   char timbuf [24];
    struct dsc$descriptor_s dsc = {
        23, DSC$K_DTYPE_T, DSC$K_CLASS_S, timbuf};
    FILE *fp = fopen ("datatypes.dat", "r", "ctx=bin");

    memset (&timbuf, 0, sizeof(timbuf));

    while (fread (&lm, sizeof(lm), 1, fp) != 0)
    {
        printf ("Action     = %c\n",     lm.rdb$lm_action);
        printf ("Table      = %.*s\n",   sizeof(lm.rdb$lm_relation_name),
                                         lm.rdb$lm_relation_name);
        printf ("Type       = %d\n",     lm.rdb$lm_record_type);
        printf ("Data Len   = %d\n",     lm.rdb$lm_data_len);
        printf ("Null Bits  = %d\n",     lm.rdb$lm_nbv_len);

        memcpy (&dbkey, &lm.rdb$lm_dbk, sizeof(lm.rdb$lm_dbk));
        printf ("DBKEY      = %d:%d:%d\n", dbkey.dbid,
                                        dbkey.pno,
                                        dbkey.lno);

        sys$asctim (0, &dsc, &lm.rdb$lm_start_tad, 0);
        printf ("Start TAD  = %s\n", timbuf);

        sys$asctim (0, &dsc, &lm.rdb$lm_commit_tad, 0);
        printf ("Commit TAD = %s\n", timbuf);

        printf ("TSN        = %Ld\n",    lm.rdb$lm_tsn);
        printf ("Version    = %d\n",     lm.rdb$lm_record_version);

        if (lm.nbv.n_tinyint == 0)
                printf ("f_tinyint  = %d\n", lm.f_tinyint);
        else    printf ("f_tinyint  = NULL\n");

        if (lm.nbv.n_smallint == 0)
                printf ("f_smallint = %d\n", lm.f_smallint);
        else    printf ("f_smallint = NULL\n");

        if (lm.nbv.n_integer == 0)
                printf ("f_integer  = %d\n", lm.f_integer);
        else    printf ("f_integer  = NULL\n");
```

```
        if (lm.nbv.n_bigint == 0)
                printf ("f_bigint   = %Ld\n", lm.f_bigint);
        else    printf ("f_bigint   = NULL\n");

        if (lm.nbv.n_double == 0)
                printf ("f_double   = %f\n", lm.f_double);
        else    printf ("f_double   = NULL\n");

        if (lm.nbv.n_real == 0)
                printf ("f_real     = %f\n", lm.f_real);
        else    printf ("f_real     = NULL\n");

        if (lm.nbv.n_fixstr == 0)
                printf ("f_fixstr   = %.*s\n", sizeof (lm.f_fixstr),
                                               lm.f_fixstr);
        else    printf ("f_fixstr   = NULL\n");

        if (lm.nbv.n_varstr == 0)
                printf ("f_varstr   = %.*s\n", lm.f_varstr_len, lm.f_varstr);
        else    printf ("f_varstr   = NULL\n");

        printf ("\n");
    }
}
```

Example sequence of commands to create a table, unload the data and display the contents with this program:

```
SQL> ATTACH 'FILE MF_PERSONNEL';
SQL> CREATE TABLE DATATYPES (
       F_TINYINT TINYINT
      ,F_SMALLINT SMALLINT
      ,F_INTEGER INTEGER
      ,F_BIGINT BIGINT
      ,F_DOUBLE DOUBLE PRECISION
      ,F_REAL REAL
      ,F_FIXSTR CHAR (10)
      ,F_VARSTR VARCHAR (10));
SQL> COMMIT;
SQL> INSERT INTO DATATYPES VALUES (1, NULL, 2, NULL, 3, NULL, 'THIS', NULL);
SQL> INSERT INTO DATATYPES VALUES (NULL, 4, NULL, 5, NULL, 6, NULL, 'THAT');
SQL> COMMIT;
SQL> EXIT;
$ RMU /BACKUP /AFTER_JOURNAL MF_PERSONNEL AIJBCK.AIJ
$ RMU /UNLOAD /AFTER_JOURNAL MF_PERSONNEL AIJBCK.AIJ -
     /TABLE = (NAME=DATATYPES, OUTPUT=DATATYPES.DAT)
$ CC DATATYPES.C
$ LINK DATATYPES.OBJ
$ RUN DATATYPES.EXE
```

## 8.9.2 New Transaction_Mode Qualifier for Oracle RMU Commands

A new qualifier, Transaction_Mode, has been added to the RMU Copy, Move_Area, Restore, and Restore Only_Root commands. You can use this qualifier to set the allowable transaction modes for the database root file created by these commands. If you are not creating a root file as part of one of these commands, for example, you are restoring an area, attempting to use this qualifier returns a CONFLSWIT error. This qualifier is similar to the SET TRANSACTION MODE clause of the CREATE DATABASE command in interactive SQL.

The primary use of this qualifier is when you restore a backup file (of the master database) to create a Hot Standby database. Include the Transaction_Mode qualifier on the RMU Restore command when you create the standby database (prior to starting replication operations). Because only read−only transactions are allowed on the standby database, you should use the Transaction_Mode=Read_Only qualifier setting. This setting prevents modifications to the standby database at all times, even when replication operations are not active.

You can specify the following transaction modes for the Transaction_Mode qualifier:

```
All
Current
None
[No]Batch_Update
[No]Read_Only
[No]Exclusive
[No]Exclusive_Read
[No]Exclusive_Write
[No]Protected
[No]Protected_Read
[No]Protected_Write
[No]Shared
[No]Shared_Read
[No]Shared_Write
```

Note that [No] indicates that the value can be negated. For example, the NoExclusive_Write option indicates that exclusive write is not an allowable access mode for this database. If you specify the Shared, Exclusive, or Protected option, Oracle RMU assumes you are referring to both reading and writing in these modes. For example, the Transaction_Mode=Shared option indicates that you want both Shared_Read and Shared_Write as transaction modes. No mode is enabled unless you add that mode to the list or you use the ALL option to enable all modes.

You cannot negate the following three options: All, which enables all transaction modes; None, which disables all transaction modes; and Current, which enables all transaction modes that are set for the source database. If you do not specify the Transaction_Mode qualifier, Oracle RMU uses the transaction modes enabled for the source database.

You can list one qualifier that enables or disables a particular mode followed by another that does the opposite. For example, Transaction_Mode=(NoShared_Write, Shared) is ambiguous because the first value disables Shared_Write access while the second value enables Shared_Write access. Oracle RMU resolves the ambiguities by first enabling all modes that are enabled by the items in the Transaction_Mode list and then disabling those modes that are disabled by items in the Transaction_Mode list. The order of items in the list is irrelevant. In the example discussed, Shared_Read is enabled and Shared_Write is disabled.

The following example shows how to set a newly restored database to allow read−only transactions only. After Oracle RMU executes the command, the database is ready for you to start Hot Standby replication operations.

```
$ RMU/RESTORE/TRANSACTION_MODE=READ_ONLY MF_PERSONNEL.RBF
```

### 8.9.3 RMU Server After_Journal Stop Command

If database replication is active and you attempt to stop the database AIJ Log Server, Oracle Rdb returns an error. You must stop database replication before attempting to stop the server.

In addition, a new qualifier, Output=filename, has been added to the RMU Server After_Journal Stop command. This optional qualifier allows you to specify the file where the operational log is to be created. The operational log records the transmission and receipt of network messages.

If you do not include a directory specification with the file name, the log file is created in the database root file directory. It is invalid to include a node name as part of the file name specification.

Note that all Hot Standby bugcheck dumps are written to the corresponding bugcheck dump file; bugcheck dumps are not written to the file you specify with the Output qualifier.

### 8.9.4 Incomplete Description of Protection Qualifier for RMU Backup After_Journal Command

The description of the Protection Qualifier for the RMU Backup After_Journal command is incomplete in the Oracle RMU Reference Manual for Digital UNIX. The complete description is as follows:

The Protection qualifier specifies the system file protection for the backup file produced by the RMU Backup After_Journal command. If you do not specify the Protection qualifier, the default access permissions are −rw−r−−−−− for backups to disk or tape.

Tapes do not allow delete or execute access and the superuser account always has both read and write access to tapes. In addition, a more restrictive class accumulates the access rights of the less restrictive classes.

If you specify the Protection qualifier explicitly, the differences in access permissions applied for backups to tape or disk as noted in the preceding paragraph are applied. Thus, if you specify Protection=(S,O,G:W,W:R), the access permissions on tape becomes rw−rw−r−.

### 8.9.5 RMU Extract Command Options Qualifier

A documentation error exists in the description of the Options=options−list qualifier of the RMU Extract command. Currently, the documentation states that this qualifier is not applied to output created by the Items=Volume qualifier. This is incorrect. Beginning with 6.1 of Oracle Rdb, the behavior of the Options=options−list qualifier is applied to output created by the Items=Volume qualifier.

### 8.9.6 RDM$SNAP_QUIET_POINT Logical is Incorrect

On page 2−72 of the Oracle RMU Reference Manual, the reference to the RDM$SNAP_QUIET_POINT logical is incorrect. The correct logical name is

RDM$BIND_SNAP_QUIET_POINT.

### 8.9.7 Using Delta Time with RMU Show Statistics Command

Oracle RMU does not support the use of delta time. However, because the OpenVMS platform does, there is a workaround. You can specify delta time using the following syntax with the RMU Show Statistics command:

```
$ RMU/SHOW STATISTICS/OUTPUT=file-spec/UNTIL=" ' ' f$cvtime ("+7:00") ' "
```

The +7:00 adds 7 hours to the current time.

You can also use "TOMORROW" and "TODAY+n".

This information will be added to the description of the Until qualifier of the RMU Show Statistics command in a future release of the Oracle RMU Reference Manual.

# 8.10 Oracle Rdb7 Guide to Database Performance and Tuning

The following section provides corrected, clarified, or omitted information for the Oracle Rdb7 Guide to Database Performance and Tuning manual.

## 8.10.1 Dynamic OR Optimization Formats

In Table C−2 on Page C−7 of the Oracle Rdb7 Guide to Database Performance and Tuning, the dynamic OR optimization format is incorrectly documented as [l:h...]n. The correct formats for Oracle Rdb Release 7.0 and later are [(l:h)n] and [l:h,l2:h2].

## 8.10.2 Oracle Rdb Logical Names

The Oracle Rdb7 Guide to Database Performance and Tuning contains a table in Chapter 2 summarizing the Oracle Rdb logical names. The information in the following table supersedes the entries for the RDM$BIND_RUJ_ALLOC_BLKCNT and RDM$BIND_RUJ_EXTEND_BLKCNT logical names.

RDM$BIND_RUJ_ALLOC_BLKCNT Allows you to override the default value of the .ruj file. The block count value can be defined between 0 and 2 billion with a default of 127.

RDM$BIND_RUJ_EXTEND_BLKCNT Allows you to pre−extend the .ruj files for each process using a database. The block count value can be defined between 0 and 65535 with a default of 127.

## 8.10.3 Waiting for Client Lock Message

The Oracle Rdb7 Guide to Database Performance and Tuning contains a section in Chapter 3 that describes the Performance Monitor Stall Messages screen. The section contains a list describing the "Waiting for" messages. The description of the "waiting for client lock" message was missing from the list.

A client lock indicates that an Rdb metadata lock is in use. The term client indicates that Rdb is a client of the Rdb locking services. The metadata locks are used to guarantee memory copies of the metadata (table, index and column definitions) are consistent with the on−disk versions.

The "waiting for client lock" message means the database user is requesting an incompatible locking mode. For example, when trying to drop a table which is in use, the drop operation requests a PROTECTED WRITE lock on the metadata object (such as a table) which is incompatible with the existing PROTECTED READ lock currently used by other users of the table.

The lock name for these special locks consist of an encoded 16 byte name. The first 4 bytes contains the leading four bytes of the user name (for system objects the RDB$ prefix is skipped) followed by three longwords. The lock is displayed in text format first − here will be seen the prefix for the table, routine, or module name; followed by its hexadecimal representation. The text version masks out non−printable characters with a dot (.).

```
waiting for client '...."...EMPL' 4C504D450000002200000000400000055
```

The leftmost value seen in the hexadecimal output contains the name prefix which is easier read in the text field. Then comes a hex number (00000022) which is the id of the object. The id is described below for tables, views, functions, procedures, modules, and sequences.

- For tables and views, the id represents the unique value found in the RDB$RELATION_ID column of the RDB$RELATIONS system relation for the given table.
- For routines (that is functions and procedures), the id represents the unique value found in the RDB$ROUTINE_ID column of the RDB$ROUTINES system relation for the given routine.
- For modules, the id represents the unique value found in the RDB$MODULE_ID column of the RDB$MODULES system relation for the given module.
- For sequences, the id represents the unique value found in the RDB$SEQUENCE_ID column of the RDB$SEQUENCES system relation for the given sequence.

The next value displayed signifies the object type. The following table describes objects and their hexadecimal type values.

***Table 8−2 Objects and Their Hexadecimal Type Value***

| Object | Hexadecimal Value |
|---|---|
| Tables or views | 00000004 |
| Modules | 00000015 |
| Routines | 00000016 |
| Sequences | 00000019 |

The last value in the hexadecimal output represents the lock type. The hexadecimal value 55 indicates this is a client lock and distinct from page and other data structure locks.

The following example shows a "waiting for client lock" message from a Stall Messages screen while the application was processing the EMPLOYEES table from MF_PERSONNEL. The terminal should be set to 132 characters wide to view the full client lock string.

```
Process.ID  Since.................. T Stall.reason.............................Lock.ID.
27800643:1                            waiting for logical area 79 (CW)         16004833
27800507:1  31−OCT−2002 16:05:15.71 W waiting for client '...."...EMPL' 4C504D45000000220
```

To determine the name of the referenced object given the lock ID, the following queries can be used based on the object type:

```
SQL> select RDB$RELATION_NAME from RDB$RELATIONS where RDB$RELATION_ID = 25;
SQL> select RDB$MODULE_NAME from RDB$MODULES where RDB$MODULE_ID = 12;
SQL> select RDB$ROUTINE_NAME from RDB$ROUTINES where RDB$ROUTINE_ID = 7;
SQL> select RDB$SEQUENCE_NAME from RDB$SEQUENCES where RDB$SEQUENCE_ID = 2;
```

For more detailed lock information, perform the following steps:

- Press the L option from the horizontal menu to display a menu of lock IDs.
- Select the desired lock ID.

## 8.10.4 RDMS$TTB_HASH_SIZE Logical Name

The logical name RDMS$TTB_HASH_SIZE sets the size of the hash table used for temporary tables. If the logical name is not defined, Oracle Rdb uses a default value of 1249.

If you expect that temporary tables will be large (that is, 10K or more rows), use this logical name to adjust the hash table size to avoid long hash chains. Set the value to approximately 1/4 of the expected maximum number of rows for each temporary table. For example, if a temporary table will be populated with 100,000 rows, define this logical name to be 25000. If there are memory constraints on your system, you should define the logical name to be no higher than this value (1/4 of the expected maximum number of rows).

## 8.10.5 Error in Updating and Retrieving a Row by Dbkey Example 3−22

Example 3−22 in Section 3.8.3 that shows how to update and retrieve a row by dbkey is incorrect. The example should appear as follows:

```
SQL> ATTACH 'FILENAME MF_PERSONNEL.RDB';
SQL> --
SQL> -- Declare host variables
SQL> --
SQL> DECLARE :hv_row INTEGER;              -- Row counter
SQL> DECLARE :hv_employee_id ID_DOM;       -- EMPLOYEE_ID field
SQL> DECLARE :hv_employee_id_ind SMALLINT; -- Null indicator variable
SQL> --
SQL> DECLARE :hv_dbkey CHAR(8);            -- DBKEY storage
SQL> DECLARE :hv_dbkey_ind SMALLINT;       -- Null indicator variable
SQL> --
SQL> DECLARE :hv_last_name LAST_NAME_DOM;
SQL> DECLARE :hv_new_address_data_1 ADDRESS_DATA_1_DOM;
SQL> --
SQL> SET TRANSACTION READ WRITE;
SQL> BEGIN
cont> --
cont> -- Set the search value for SELECT
cont> --
cont>  SET :hv_last_name = 'Ames';
cont> --
cont> -- Set the NEW_ADDRESS_DATA_1 value
cont> --
cont> SET :hv_new_address_data_1 = '100 Broadway Ave.';
cont> END;
SQL> COMMIT;
SQL> --
SQL> SET TRANSACTION READ ONLY;
SQL> BEGIN
cont> SELECT E.EMPLOYEE_ID, E.DBKEY
cont>   INTO :hv_employee_id INDICATOR :hv_employee_id_ind,
cont>        :hv_dbkey INDICATOR :hv_dbkey_ind
cont>   FROM EMPLOYEES E
cont> WHERE E.LAST_NAME = :hv_last_name
cont> LIMIT TO 1 ROW;
cont> --
cont> GET DIAGNOSTICS :hv_row = ROW_COUNT;
cont> END;
```

```
SQL> COMMIT;
SQL> --
SQL> SET TRANSACTION READ WRITE RESERVING EMPLOYEES FOR SHARED WRITE;
SQL> BEGIN
cont> IF (:hv_row = 1) THEN
cont>    BEGIN
cont>    UPDATE EMPLOYEES E
cont>      SET E.ADDRESS_DATA_1 = :hv_new_address_data_1
cont>    WHERE E.DBKEY = :hv_dbkey;
cont>    END;
cont> END IF;
cont> END;
SQL> COMMIT;
SQL> --
SQL> -- Display result of change
SQL> --
SQL> SET TRANSACTION READ ONLY;
SQL> SELECT E.*
cont> FROM EMPLOYEES E
cont> WHERE E.DBKEY = :hv_dbkey;
 EMPLOYEE_ID   LAST_NAME        FIRST_NAME   MIDDLE_INITIAL
   ADDRESS_DATA_1              ADDRESS_DATA_2         CITY
      STATE   POSTAL_CODE   SEX    BIRTHDAY      STATUS_CODE
 00416         Ames             Louie        A
   100 Broadway Ave.                                  Alton
      NH      03809         M      13-Apr-1941   1

1 row selected
SQL>
```

The new example will appear in a future publication of the Oracle Rdb7 Guide to Database Performance and Tuning manual.

## 8.10.6 Error in Calculation of Sorted Index in Example 3−46

Example 3−46 in Section 3.9.5.1 shows the output when you use the RMU Analyze Indexes command and specify the Option=Debug qualifier and the DEPARTMENTS_INDEX sorted index.

The description of the example did not include the 8 byte dbkey in the calculation of the sorted index. The complete description is as follows:

The entire index (26 records) is located on pages 2 and 3 in logical area 72 and uses 188 bytes of a possible 430 bytes or the node record is 47 percent full. Note that due to index compression, the node size has decreased in size from 422 bytes to 188 bytes and the percent fullness of the node records has dropped from 98 to 47 percent. Also note that the used/avail value in the summary information at the end of the output does not include the index header and trailer information, which accounts for 32 bytes. This value is shown for each node record in the detailed part of the output. The number of bytes used by the index is calculated as follows: the sort key is 4 bytes plus a null byte for a total of 5 bytes. The prefix is 1 byte and the suffix is 1 byte. The prefix indicates the number of bytes in the preceding key that are the same and the suffix indicates the number of bytes that are different from the preceding key. The dbkey pointer to the row is 8 bytes. There are 26 data rows multiplied by 15 bytes for a total of 390 bytes. The 15 bytes include:

- ♦ 7 bytes for the sort key: length + null byte + prefix + suffix
- ♦ 8 bytes for the dbkey pointer to the row

Add 32 bytes for index header and trailer information for the index node to the 390 bytes for a total of 422 bytes used. Index compression reduces the number of bytes used to 188 bytes used.

The revised description will appear in a future publication of the Oracle Rdb7 Guide to Database Performance and Tuning manual.

## 8.10.7 Documentation Error in Section C.7

The Oracle Rdb Guide to Database Performance And Tuning, Volume 2 contains an error in Section C.7 titled Displaying Sort Statistics with the R Flag.

When describing the output from this debugging flag, bullet 9 states:

- Work File Alloc indicates how many work files were used in the sort operation. A zero (0) value indicates that the sort was accomplished completely in memory.

This is incorrect, the statistics should be described as show below:

- Work File Alloc indicates how much space (in blocks) was allocated in the work files for this sort operation. A zero (0) value indicates that the sort was accomplished completely in memory.

This error will be corrected in a future release of Oracle Rdb Guide to Database Performance And Tuning.

## 8.10.8 Missing Tables Descriptions for the RDBEXPERT Collection Class

Appendix B in the Oracle Rdb7 Guide to Database Performance and Tuning describes the event−based data tables in the formatted database for the Oracle Rdb PERFORMANCE and RDBEXPERT collection classes. This section describes the missing tables for the RDBEXPERT collection class.

Table 8−3 shows the TRANS_TPB table.

***Table 8−3 Columns for Table EPC$1_221_TRANS_TPB***

| Column Name | Data Type | Domain |
|---|---|---|
| COLLECTION_RECORD_ID | SMALLINT | COLLECTION_RECORD_ID_DOMAIN |
| IMAGE_RECORD_ID | INTEGER | IMAGE_RECORD_ID_DOMAIN |
| CONTEXT_NUMBER | INTEGER | CONTEXT_NUMBER_DOMAIN |
| TIMESTAMP_POINT | DATE VMS | |
| CLIENT_PC | INTEGER | |
| STREAM_ID | INTEGER | |
| TRANS_ID | VARCHAR(16) | |
| TRANS_ID_STR_ID | INTEGER | STR_ID_DOMAIN |
| TPB | VARCHAR(127) | |
| TPB_STR_ID | INTEGER | STR_ID_DOMAIN |

Table 8−4 shows the TRANS_TPB_ST table. An index is provided for this table. It is defined with column STR_ID, duplicates are allowed, and the type is sorted.

***Table 8−4 Columns for Table EPC$1_221_TRANS_TPB_ST***

| Column Name | Data Type | Domain |
|---|---|---|
| STR_ID | INTEGER | STR_ID_DOMAIN |
| SEGMENT_NUMBER | SMALLINT | SEGMENT_NUMBER_DOMAIN |
| STR_SEGMENT | VARCHAR(128) | |

## 8.10.9 Missing Columns Descriptions for Tables in the Formatted Database

Some of the columns were missing from the tables in Appendix B in the Oracle Rdb7 Guide to Database Performance and Tuning. The complete table definitions are described in this section.

Table 8−5 shows the DATABASE table.

***Table 8−5 Columns for Table EPC$1_221_DATABASE***

| Column Name | Data Type | Domain |
|---|---|---|
| COLLECTION_RECORD_ID | SMALLINT | COLLECTION_RECORD_ID_DOMAIN |
| IMAGE_RECORD_ID | INTEGER | IMAGE_RECORD_ID_DOMAIN |
| CONTEXT_NUMBER | INTEGER | CONTEXT_NUMBER_DOMAIN |
| TIMESTAMP_POINT | DATE VMS | |
| CLIENT_PC | INTEGER | |
| STREAM_ID | INTEGER | |
| DB_NAME | VARCHAR(255) | |
| DB_NAME_STR_ID | INTEGER | STR_ID_DOMAIN |
| IMAGE_FILE_NAME | VARCHAR(255) | |
| IMAGE_FILE_NAME_STR_ID | INTEGER | STR_ID_DOMAIN |

Table 8−6 shows the REQUEST_ACTUAL table.

***Table 8−6 Columns for Table EPC$1_221_REQUEST_ACTUAL***

| Column Name | Data Type | Domain |
|---|---|---|
| COLLECTION_RECORD_ID | SMALLINT | COLLECTION_RECORD_ID_DOMAIN |
| IMAGE_RECORD_ID | INTEGER | IMAGE_RECORD_ID_DOMAIN |
| CONTEXT_NUMBER | INTEGER | CONTEXT_NUMBER_DOMAIN |
| TIMESTAMP_START | DATE VMS | |

| | |
|---|---|
| TIMESTAMP_END | DATE VMS |
| DBS_READS_START | INTEGER |
| DBS_WRITES_START | INTEGER |
| RUJ_READS_START | INTEGER |
| RUJ_WRITES_START | INTEGER |
| AIJ_WRITES_START | INTEGER |
| ROOT_READS_START | INTEGER |
| ROOT_WRITES_START | INTEGER |
| BUFFER_READS_START | INTEGER |
| GET_VM_BYTES_START | INTEGER |
| FREE_VM_BYTES_START | INTEGER |
| LOCK_REQS_START | INTEGER |
| REQ_NOT_QUEUED_START | INTEGER |
| REQ_STALLS_START | INTEGER |
| REQ_DEADLOCKS_START | INTEGER |
| PROM_DEADLOCKS_START | INTEGER |
| LOCK_RELS_START | INTEGER |
| LOCK_STALL_TIME_START | INTEGER |
| D_FETCH_RET_START | INTEGER |
| D_FETCH_UPD_START | INTEGER |
| D_LB_ALLOK_START | INTEGER |
| D_LB_GBNEEDLOCK_START | INTEGER |
| D_LB_NEEDLOCK_START | INTEGER |
| D_LB_OLDVER_START | INTEGER |
| D_GB_NEEDLOCK_START | INTEGER |
| D_GB_OLDVER_START | INTEGER |
| D_NOTFOUND_IO_START | INTEGER |
| D_NOTFOUND_SYN_START | INTEGER |
| S_FETCH_RET_START | INTEGER |
| S_FETCH_UPD_START | INTEGER |
| S_LB_ALLOK_START | INTEGER |
| S_LB_GBNEEDLOCK_START | INTEGER |
| S_LB_NEEDLOCK_START | INTEGER |
| S_LB_OLDVER_START | INTEGER |
| S_GB_NEEDLOCK_START | INTEGER |
| S_GB_OLDVER_START | INTEGER |
| S_NOTFOUND_IO_START | INTEGER |
| S_NOTFOUND_SYN_START | INTEGER |
| D_ASYNC_FETCH_START | INTEGER |
| S_ASYNC_FETCH_START | INTEGER |
| D_ASYNC_READIO_START | INTEGER |
| S_ASYNC_READIO_START | INTEGER |

| | | |
|---|---|---|
| AS_READ_STALL_START | INTEGER | |
| AS_BATCH_WRITE_START | INTEGER | |
| AS_WRITE_STALL_START | INTEGER | |
| BIO_START | INTEGER | |
| DIO_START | INTEGER | |
| PAGEFAULTS_START | INTEGER | |
| PAGEFAULT_IO_START | INTEGER | |
| CPU_START | INTEGER | |
| CURRENT_PRIO_START | SMALLINT | |
| VIRTUAL_SIZE_START | INTEGER | |
| WS_SIZE_START | INTEGER | |
| WS_PRIVATE_START | INTEGER | |
| WS_GLOBAL_START | INTEGER | |
| CLIENT_PC_END | INTEGER | |
| STREAM_ID_END | INTEGER | |
| REQ_ID_END | INTEGER | |
| COMP_STATUS_END | INTEGER | |
| REQUEST_OPER_END | INTEGER | |
| TRANS_ID_END | VARCHAR(16) | |
| TRANS_ID_END_STR_ID | INTEGER | STR_ID_DOMAIN |
| DBS_READS_END | INTEGER | |
| DBS_WRITES_END | INTEGER | |
| RUJ_READS_END | INTEGER | |
| RUJ_WRITES_END | INTEGER | |
| AIJ_WRITES_END | INTEGER | |
| ROOT_READS_END | INTEGER | |
| ROOT_WRITES_END | INTEGER | |
| BUFFER_READS_END | INTEGER | |
| GET_VM_BYTES_END | INTEGER | |
| FREE_VM_BYTES_END | INTEGER | |
| LOCK_REQS_END | INTEGER | |
| REQ_NOT_QUEUED_END | INTEGER | |
| REQ_STALLS_END | INTEGER | |
| REQ_DEADLOCKS_END | INTEGER | |
| PROM_DEADLOCKS_END | INTEGER | |
| LOCK_RELS_END | INTEGER | |
| LOCK_STALL_TIME_END | INTEGER | |
| D_FETCH_RET_END | INTEGER | |
| D_FETCH_UPD_END | INTEGER | |
| D_LB_ALLOK_END | INTEGER | |
| D_LB_GBNEEDLOCK_END | INTEGER | |
| D_LB_NEEDLOCK_END | INTEGER | |

| | |
|---|---|
| D_LB_OLDVER_END | INTEGER |
| D_GB_NEEDLOCK_END | INTEGER |
| D_GB_OLDVER_END | INTEGER |
| D_NOTFOUND_IO_END | INTEGER |
| D_NOTFOUND_SYN_END | INTEGER |
| S_FETCH_RET_END | INTEGER |
| S_FETCH_UPD_END | INTEGER |
| S_LB_ALLOK_END | INTEGER |
| S_LB_GBNEEDLOCK_END | INTEGER |
| S_LB_NEEDLOCK_END | INTEGER |
| S_LB_OLDVER_END | INTEGER |
| S_GB_NEEDLOCK_END | INTEGER |
| S_GB_OLDVER_END | INTEGER |
| S_NOTFOUND_IO_END | INTEGER |
| S_NOTFOUND_SYN_END | INTEGER |
| D_ASYNC_FETCH_END | INTEGER |
| S_ASYNC_FETCH_END | INTEGER |
| D_ASYNC_READIO_END | INTEGER |
| S_ASYNC_READIO_END | INTEGER |
| AS_READ_STALL_END | INTEGER |
| AS_BATCH_WRITE_END | INTEGER |
| AS_WRITE_STALL_END | INTEGER |
| BIO_END | INTEGER |
| DIO_END | INTEGER |
| PAGEFAULTS_END | INTEGER |
| PAGEFAULT_IO_END | INTEGER |
| CPU_END | INTEGER |
| CURRENT_PRIO_END | SMALLINT |
| VIRTUAL_SIZE_END | INTEGER |
| WS_SIZE_END | INTEGER |
| WS_PRIVATE_END | INTEGER |
| WS_GLOBAL_END | INTEGER |

Table 8−7 shows the TRANSACTION table.

***Table 8−7 Columns for Table EPC$1_221_TRANSACTION***

| Column Name | Data Type | Domain |
|---|---|---|
| COLLECTION_RECORD_ID | SMALLINT | COLLECTION_RECORD_ID_DOMAIN |
| IMAGE_RECORD_ID | INTEGER | IMAGE_RECORD_ID_DOMAIN |
| CONTEXT_NUMBER | INTEGER | CONTEXT_NUMBER_DOMAIN |
| TIMESTAMP_START | DATE VMS | |

| | | |
|---|---|---|
| TIMESTAMP_END | DATE VMS | |
| CLIENT_PC_START | INTEGER | |
| STREAM_ID_START | INTEGER | |
| LOCK_MODE_START | INTEGER | |
| TRANS_ID_START | VARCHAR(16) | |
| TRANS_ID_START_STR_ID | INTEGER | STR_ID_DOMAIN |
| GLOBAL_TID_START | VARCHAR(16) | |
| GLOBAL_TID_START_STR_ID | INTEGER | STR_ID_DOMAIN |
| DBS_READS_START | INTEGER | |
| DBS_WRITES_START | INTEGER | |
| RUJ_READS_START | INTEGER | |
| RUJ_WRITES_START | INTEGER | |
| AIJ_WRITES_START | INTEGER | |
| ROOT_READS_START | INTEGER | |
| ROOT_WRITES_START | INTEGER | |
| BUFFER_READS_START | INTEGER | |
| GET_VM_BYTES_START | INTEGER | |
| FREE_VM_BYTES_START | INTEGER | |
| LOCK_REQS_START | INTEGER | |
| REQ_NOT_QUEUED_START | INTEGER | |
| REQ_STALLS_START | INTEGER | |
| REQ_DEADLOCKS_START | INTEGER | |
| PROM_DEADLOCKS_START | INTEGER | |
| LOCK_RELS_START | INTEGER | |
| LOCK_STALL_TIME_START | INTEGER | |
| D_FETCH_RET_START | INTEGER | |
| D_FETCH_UPD_START | INTEGER | |
| D_LB_ALLOK_START | INTEGER | |
| D_LB_GBNEEDLOCK_START | INTEGER | |
| D_LB_NEEDLOCK_START | INTEGER | |
| D_LB_OLDVER_START | INTEGER | |
| D_GB_NEEDLOCK_START | INTEGER | |
| D_GB_OLDVER_START | INTEGER | |
| D_NOTFOUND_IO_START | INTEGER | |
| D_NOTFOUND_SYN_START | INTEGER | |
| S_FETCH_RET_START | INTEGER | |
| S_FETCH_UPD_START | INTEGER | |
| S_LB_ALLOK_START | INTEGER | |
| S_LB_GBNEEDLOCK_START | INTEGER | |
| S_LB_NEEDLOCK_START | INTEGER | |
| S_LB_OLDVER_START | INTEGER | |
| S_GB_NEEDLOCK_START | INTEGER | |

| | | |
|---|---|---|
| S_GB_OLDVER_START | INTEGER | |
| S_NOTFOUND_IO_START | INTEGER | |
| S_NOTFOUND_SYN_START | INTEGER | |
| D_ASYNC_FETCH_START | INTEGER | |
| S_ASYNC_FETCH_START | INTEGER | |
| D_ASYNC_READIO_START | INTEGER | |
| S_ASYNC_READIO_START | INTEGER | |
| AS_READ_STALL_START | INTEGER | |
| AS_BATCH_WRITE_START | INTEGER | |
| AS_WRITE_STALL_START | INTEGER | |
| AREA_ITEMS_START | VARCHAR(128) | |
| AREA_ITEMS_START_STR_ID | INTEGER | STR_ID_DOMAIN |
| BIO_START | INTEGER | |
| DIO_START | INTEGER | |
| PAGEFAULTS_START | INTEGER | |
| PAGEFAULT_IO_START | INTEGER | |
| CPU_START | INTEGER | |
| CURRENT_PRIO_START | SMALLINT | |
| VIRTUAL_SIZE_START | INTEGER | |
| WS_SIZE_START | INTEGER | |
| WS_PRIVATE_START | INTEGER | |
| WS_GLOBAL_START | INTEGER | |
| CROSS_FAC_2_START | INTEGER | |
| CROSS_FAC_3_START | INTEGER | |
| CROSS_FAC_7_START | INTEGER | |
| CROSS_FAC_14_START | INTEGER | |
| DBS_READS_END | INTEGER | |
| DBS_WRITES_END | INTEGER | |
| RUJ_READS_END | INTEGER | |
| RUJ_WRITES_END | INTEGER | |
| AIJ_WRITES_END | INTEGER | |
| ROOT_READS_END | INTEGER | |
| ROOT_WRITES_END | INTEGER | |
| BUFFER_READS_END | INTEGER | |
| GET_VM_BYTES_END | INTEGER | |
| FREE_VM_BYTES_END | INTEGER | |
| LOCK_REQS_END | INTEGER | |
| REQ_NOT_QUEUED_END | INTEGER | |
| REQ_STALLS_END | INTEGER | |
| REQ_DEADLOCKS_END | INTEGER | |
| PROM_DEADLOCKS_END | INTEGER | |
| LOCK_RELS_END | INTEGER | |

| | | |
|---|---|---|
| LOCK_STALL_TIME_END | INTEGER | |
| D_FETCH_RET_END | INTEGER | |
| D_FETCH_UPD_END | INTEGER | |
| D_LB_ALLOK_END | INTEGER | |
| D_LB_GBNEEDLOCK_END | INTEGER | |
| D_LB_NEEDLOCK_END | INTEGER | |
| D_LB_OLDVER_END | INTEGER | |
| D_GB_NEEDLOCK_END | INTEGER | |
| D_GB_OLDVER_END | INTEGER | |
| D_NOTFOUND_IO_END | INTEGER | |
| D_NOTFOUND_SYN_END | INTEGER | |
| S_FETCH_RET_END | INTEGER | |
| S_FETCH_UPD_END | INTEGER | |
| S_LB_ALLOK_END | INTEGER | |
| S_LB_GBNEEDLOCK_END | INTEGER | |
| S_LB_NEEDLOCK_END | INTEGER | |
| S_LB_OLDVER_END | INTEGER | |
| S_GB_NEEDLOCK_END | INTEGER | |
| S_GB_OLDVER_END | INTEGER | |
| S_NOTFOUND_IO_END | INTEGER | |
| S_NOTFOUND_SYN_END | INTEGER | |
| D_ASYNC_FETCH_END | INTEGER | |
| S_ASYNC_FETCH_END | INTEGER | |
| D_ASYNC_READIO_END | INTEGER | |
| S_ASYNC_READIO_END | INTEGER | |
| AS_READ_STALL_END | INTEGER | |
| AS_BATCH_WRITE_END | INTEGER | |
| AS_WRITE_STALL_END | INTEGER | |
| AREA_ITEMS_END | VARCHAR(128) | |
| AREA_ITEMS_END_STR_ID | INTEGER | STR_ID_DOMAIN |
| BIO_END | INTEGER | |
| DIO_END | INTEGER | |
| PAGEFAULTS_END | INTEGER | |
| PAGEFAULT_IO_END | INTEGER | |
| CPU_END | INTEGER | |
| CURRENT_PRIO_END | SMALLINT | |
| VIRTUAL_SIZE_END | INTEGER | |
| WS_SIZE_END | INTEGER | |
| WS_PRIVATE_END | INTEGER | |
| WS_GLOBAL_END | INTEGER | |
| CROSS_FAC_2_END | INTEGER | |
| CROSS_FAC_3_END | INTEGER | |

| | |
|---|---|
| CROSS_FAC_7_END | INTEGER |
| CROSS_FAC_14_END | INTEGER |

Table 8−8 shows the REQUEST_BLR table.

***Table 8−8 Columns for Table EPC$1_221_REQUEST_BLR***

| Column Name | Data Type | Domain |
|---|---|---|
| COLLECTION_RECORD_ID | SMALLINT | COLLECTION_RECORD_ID_DOMAIN |
| IMAGE_RECORD_ID | INTEGER | IMAGE_RECORD_ID_DOMAIN |
| CONTEXT_NUMBER | INTEGER | CONTEXT_NUMBER_DOMAIN |
| TIMESTAMP_POINT | DATE VMS | |
| CLIENT_PC | INTEGER | |
| STREAM_ID | INTEGER | |
| REQ_ID | INTEGER | |
| TRANS_ID | VARCHAR(16) | |
| TRANS_ID_STR_ID | INTEGER | STR_ID_DOMAIN |
| REQUEST_NAME | VARCHAR(31) | |
| REQUEST_NAME_STR_ID | INTEGER | STR_ID_DOMAIN |
| REQUEST_TYPE | INTEGER | |
| BLR | VARCHAR(127) | |
| BLR_STR_ID | INTEGER | STR_ID_DOMAIN |

## 8.10.10 A Way to Find the Transaction Type of a Particular Transaction Within the Trace Database

The table EPC$1_221_TRANSACTION in the formatted Oracle Trace database has a column LOCK_MODE_START of longword datatype. The values of this column indicate the type of transaction a particular transaction was.

```
Value           Transaction type
-----           ----------------
8               Read only
9               Read write
14              Batch update
```

## 8.10.11 Using Oracle TRACE Collected Data

The following example shows how the OPTIMIZE AS clause is reflected in the Oracle TRACE database. When a trace collection is started the following SQL commands will record the request names.

```
SQL> attach `file personnel';
SQL> select last_name, first_name
```

```
cont> from employees
cont> optimize as request_one;
.
.
.
SQL> select employee_id
cont> from employees
cont> optimize as request_two;
.
.
.
SQL> select employee_id, city, state
cont> from employees
cont> optimize as request_three;
.
.
.
SQL> select last_name, first_name, employee_id, city, state
cont> from employees
cont> optimize as request_four;
.
.
.
```

Once an Oracle TRACE database has been populated from the collection, a query such as the following can be used to display the request names and types. The type values are described in Table 3−10. The unnamed queries in this example correspond to the queries executed by interactive SQL to validate the names of the tables an columns referenced in the user supplied queries.

```
SQL> select REQUEST_NAME, REQUEST_TYPE, TIMESTAMP_POINT
cont> from EPC$1_221_REQUEST_BLR;
REQUEST_NAME                        REQUEST_TYPE   TIMESTAMP_POINT
                                               1   15−JAN−1997 13:23:27.18
                                               1   15−JAN−1997 13:23:27.77
REQUEST_ONE                                    1   15−JAN−1997 13:23:28.21
REQUEST_TWO                                    1   15−JAN−1997 13:23:56.55
REQUEST_THREE                                  1   15−JAN−1997 13:24:57.27
REQUEST_FOUR                                   1   15−JAN−1997 13:25:25.44
6 rows selected
```

The next example shows the internal query format (BLR) converted to SQL strings after EPC$EXAMPLES:EPC_BLR_TOSQL_CONVERTER.COM has been run.

```
SQL> SELECT A.REQUEST_NAME, B.SQL_STRING FROM
cont> EPC$1_221_REQUEST_BLR A,
cont> EPC$SQL_QUERIES B
cont> WHERE A.CLIENT_PC = 0 AND A.SQL_ID = B.SQL_ID;
A.REQUEST_NAME
  B.SQL_STRING
REQUEST_ONE
      SELECT C1.LAST_NAME, C1.FIRST_NAME.        FROM EMPLOYEES C1
. . .
REQUEST_TWO
      SELECT C1.EMPLOYEE_ID.        FROM EMPLOYEES C1
. . .
REQUEST_THREE
SELECT C1.EMPLOYEE_ID, C1.CITY, C1.STATE.        FROM EMPLOYEES C1
  .
  .
```

```
   .
4 rows selected
```

Table 4−17 shows the Request Types.

***Table 8−9 Request Types***

| Symbolic Name | Value | Comment |
|---|---|---|
| RDB_K_REQTYPE_OTHER | 0 | A query executed internally by Oracle Rdb |
| RDB_K_REQTYPE_USER_REQUEST | 1 | A non−stored SQL statement, which includes compound statements |
| RDB_K_REQTYPE_PROCEDURE | 2 | A stored procedure |
| RDB_K_REQTYPE_FUNCTION | 3 | A stored function |
| RDB_K_REQTYPE_TRIGGER | 4 | A trigger action |
| RDB_K_REQTYPE_CONSTRAINT | 5 | A table or column constraint |

## 8.10.12 AIP Length Problems in Indexes that Allow Duplicates

When an index allows duplicates, the length stored in the AIP will be 215 bytes, regardless of the actual index node size. Because an index with duplicates can have variable node sizes, the 215−byte size is used as a median length to represent the length of rows in the index's logical area.

When the row size in the AIP is less than the actual row length, it is highly likely that SPAM entries will show space is available on pages when they have insufficient space to store another full size row. This is the most common cause of insert performance problems.

For example, consider a case where an index node size of 430 bytes (a common default value) is used; the page size for the storage area where the index is stored is 2 blocks. After deducting page overhead, the available space on a 2−block page is 982 bytes. Assume that the page in this example is initially empty.

1. A full size (430−byte) index node is stored. As 8 bytes of overhead are associated with each row stored on a page, that leaves 982−430−8 = 544 free bytes remaining on the page.
2. A duplicate key entry is made in that index node and thus a duplicate node is created on the same page. An initial duplicate node is 112 bytes long (duplicate nodes can have a variety of sizes depending on when they are created, but for this particular example, 112 bytes is used). Therefore, 544−112−8 = 424 free bytes remain on the page.

At this point, 424 bytes are left on the page. That is greater than the 215 bytes that the AIP shows as the row length for the logical area, so the SPAM page shows that the page has space available. However, an attempt to store a full size index node on the page will fail, because the remaining free space (424 bytes) is not enough to store a 430−byte node.

In this case, another candidate page must be selected via the SPAM page, and the process repeats until a page that truly has sufficient free space available is found. In a logical area that contains many duplicate nodes, a significant percentage of the pages in the logical area may fit the scenario just described. When that is the case, and a new full size index node needs to be stored, many pages may need to be read and checked before one is found that can be used to store the row.

It is possible to avoid the preceding scenario by using logical area thresholds. The goal is to set a threshold such that the SPAM page will show a page is full when space is insufficient to store a full size index node.

Using the previous example, here is how to properly set logical area thresholds to prevent excessive pages checked on an index with a 430−byte node size that is stored on a 2−block page. To calculate the proper threshold value to use, you must first determine how full the page can get before no more full size nodes will fit on the page. In this example, a database page can have up to 982−430−8 = 544 bytes in use before the page is too full. Therefore, if 544 or fewer bytes are in use, then enough space remains to store another full size node. The threshold is then 544 / 982 = .553971, or 55%.

In addition, you can determine how full a page must be before a duplicate node of size 112 will no longer fit. In this example, a database page can have up to 982−112−8 = 862 bytes in use before the page is too full. Therefore, if 862 or fewer bytes are in use, then enough space remains to store another small duplicates node. The threshold is then 862 / 982 = .8778, or 88%.

Here is an example of creating an index with the above characteristics:

```
SQL> CREATE INDEX TEST_INDEX ON EMPLOYEES (LAST_NAME)
cont>     STORE IN RDB$SYSTEM
cont>     (THRESHOLD IS (55, 55, 88));
```

These settings mean that any page at over 55% full will not be fetched when inserting a full index node, however, it may be fetched when inserting the smaller duplicates node. When the page is over 88% full then neither a full node nor a duplicate node can be stored, so the page is set as FULL. The lowest setting is not used and so can be set to any value less than or equal to the lowest used threshold.

Note that the compression algorithm used on regular tables that have compression enabled does not apply to index nodes. Index nodes are not compressed like data rows and will always utilize the number of bytes that is specified in the node size. Do not attempt to take into account a compression factor when calculating thresholds for indexes.

## 8.10.13 RDM$BIND_MAX_DBR_COUNT Documentation Clarification

Appendix A in Oracle Rdb7 Guide to Database Performance and Tuning incorrectly describes the use of the RDM$BIND_MAX_DBR_COUNT logical name.

Following is an updated description. Note that the difference in actual behavior between what is in the existing documentation and the software is that the logical name only controls the number of database recovery processes created at once during "node failure" recovery (that is, after a system or monitor crash or other abnormal shutdown).

When an entire database is abnormally shut down (due, for example, to a system failure), the database will have to be recovered in a "node failure" recovery mode. This recovery will be performed by another monitor in the cluster if the database is opened on another node or will be performed the next time the database is opened.

The RDM$BIND_MAX_DBR_COUNT logical name and the RDB_BIND_MAX_DBR_COUNT configuration parameter define the maximum number of database recovery (DBR) processes to be simultaneously invoked by the database monitor during a "node failure" recovery.

This logical name and configuration parameter apply only to databases that do not have global buffers enabled. Databases that utilize global buffers have only one recovery process started at a time during a "node failure" recovery.

In a node failure recovery situation with the Row Cache feature enabled (regardless of the global buffer state), the database monitor will start a single database recovery (DBR) process to recover the Row Cache Server (RCS) process and all user processes from the oldest active checkpoint in the database.

# 8.11 Oracle Rdb7 Guide to SQL Programming

This section provides information that is missing or changed in the Oracle Rdb7 Guide to SQL Programming.

## 8.11.1 Location of Host Source File Generated by the SQL Precompiler

When the SQL precompiler generates host source files (for example, .c, .pas, or .for) from the precompiler source files, it locates these files based on the Object qualifier in the command given to the SQL precompiler.

The following examples show the location where the host source file is generated.

When the Object qualifier is not specified on the command line, the object and the host source file take the name of the SQL precompiler with the extensions of .obj and .c, respectively. For example:

```
$ sqlpre/cc scc_try_mli_successful.sc
$ dir scc_try_mli_successful.*

Directory MYDISK:[LUND]

SCC_TRY_MLI_SUCCESSFUL.C;1              SCC_TRY_MLI_SUCCESSFUL.OBJ;2
SCC_TRY_MLI_SUCCESSFUL.SC;2

Total of 3 files.
```

When the Object qualifier is specified on the command line, the object and the host source take the name given on the qualifier switch. It uses the default of the SQL precompiler source if a filespec is not specified. It uses the defaults of .obj and .c if the extension is not specified. If the host language is a language other than C, it uses the appropriate host source extension (for example, .pas or .for). The files also default to the current directory if a directory specification is not specified. For example:

```
$ sqlpre/cc/obj=myobj scc_try_mli_successful.sc
$ dir scc_try_mli_successful.*

Directory MYDISK:[LUND]

SCC_TRY_MLI_SUCCESSFUL.SC;2

Total of 1 file.
$ dir myobj.*

Directory MYDISK:[LUND]

MYOBJ.C;1           MYOBJ.OBJ;2

Total of 2 files.

$ sqlpre/cc/obj=MYDISK:[lund.tmp] scc_try_mli_successful.sc
$ dir scc_try_mli_successful.*

Directory MYDISK:[LUND]
```

```
SCC_TRY_MLI_SUCCESSFUL.SC;2

Total of 1 file.
$ dir MYDISK:[lund.tmp]scc_try_mli_successful.*

Directory MYDISK:[LUND.TMP]

SCC_TRY_MLI_SUCCESSFUL.C;1              SCC_TRY_MLI_SUCCESSFUL.OBJ;2

Total of 2 files.
```

## 8.11.2 Remote User Authentication

In the Oracle Rdb7 Guide to SQL Programming, Table 15−1 indicates that implicit authorization works from an OpenVMS platform to another OpenVMS platform using TCP/IP. This table is incorrect. Implicit authorization only works using DECnet in this situation.

The Oracle Rdb7 Guide to SQL Programming will be fixed in a future release.

## 8.11.3 Additional Information About Detached Processes

Oracle Rdb documentation omits necessary detail on running Oracle Rdb from a detached process.

Applications run from detached processes must ensure that the OpenVMS environment is established correctly before running Oracle Rdb, otherwise Oracle Rdb will not execute.

Attempts to attach to a database and execute an Oracle Rdb query from applications running as detached processes will result in an error similar to the following:

```
%RDB−F−SYS_REQUEST, error from system services request
−SORT−E−OPENOUT, error opening [file] as output
−RMS−F−DEV, error in device name or inappropriate device type for operation
```

The problem occurs because a detached process does not normally have the logical names SYS$LOGIN or SYS$SCRATCH defined.

There are two methods that can be used to correct this:

- Solution 1:
  Use the DCL command procedure RUN_PROCEDURE to run the ACCOUNTS application:
  RUN_PROCEDURE.COM includes the single line:
  $ RUN ACCOUNTS_REPORT
  Then execute this procedure using this command:
  $ RUN/DETACH/AUTHORIZE SYS$SYSTEM:LOGINOUT/INPUT=RUN_PROCEDURE
  This solution executes SYS$SYSTEM:LOGINOUT so that the command language interface (DCL) is activated. This causes the logical names SYS$LOGIN and SYS$SCRATCH to be defined for the detached process. The /AUTHORIZE qualifier also ensures that the users' process quota limits (PQLs) are used from the system authorization file rather than relying on the default PQL system parameters, which are often insufficient to run Oracle Rdb.
- Solution 2:
  If DCL is not desired, and SYS$LOGIN and SYS$SCRATCH are not defined, then prior to

executing any Oracle Rdb statement, you should define the following logical names:

◊ RDMS$BIND_WORK_FILE

  Define this logical name to allow you to reduce the overhead of disk I/O operations for matching operations when used in conjunction with the RDMS$BIND_WORK_VM logical name. If the virtual memory file is too small then overflow to disk will occur at the disk and directory location specified by RDMS$BIND_WORK_FILE.

  For more information on RDMS$BIND_WORK_FILE and RDMS$BIND_WORK_VM, see the Oracle Rdb Guide to Database Performance and Tuning.

◊ SORTWORK0, SORTWORK1, and so on

  The OpenVMS Sort/Merge utility (SORT/MERGE) attempts to create sort work files in SYS$SCRATCH. If the SORTWORK logical names exist, the utility will not require the SYS$SCRATCH logical. However, note that not all queries will require sorting, and that some sorts will be completed in memory and so will not necessarily require disk space.

  If you use the logical RDMS$BIND_SORT_WORKFILES, you will need to define further SORTWORK logical names as described in the Oracle Rdb Guide to Database Performance and Tuning.

  You should also verify that sufficient process quotas are specified on the RUN/DETACH command line, or defined as system PQL parameters to allow Oracle Rdb to execute.

# 8.12 Guide to Using Oracle SQL/Services Client APIs

The following information describes Oracle SQL/Services documentation errors or omissions.

- The Guide to Using Oracle SQL/Services Client APIs does not describe changes to size and format of integer and floating−point data types
  Beginning with Oracle SQL/Services V5.1, the size and format of some integer and floating−point data types is changed as follows:
  - Trailing zeros occur in fixed−point numeric data types with SCALE FACTOR.
    Trailing zeros are now included after the decimal point up to the number of digits specified by the SCALE FACTOR. In versions of Oracle SQL/Services previous to V5.1, at most one trailing zero was included where the value was a whole number.
    The following examples illustrate the changes using a field defined as INTEGER(3):

    ```
    V5.1 and      Versions previous
    higher        to V5.1
    --------      -----------------
      1.000         1.0
     23.400        23.4
    567.890       567.89
    ```
  - Trailing zeros occur in floating−point data types. Trailing zeros are now included in the fraction, and leading zeros are included in the exponent, up to the maximum precision available, for fields assigned the REAL and DOUBLE PRECISION data types.

    ```
                                                        Versions previous
    Data Type           V5.1 and higher                 to V5.1
    ----------------    ----------------------          -----------------
    REAL                1.2340000E+01                   1.234E+1
    DOUBLE PRECISION    5.678900000000000E+001          5.6789E+1
    ```
  - Size of TINYINT and REAL data types is changed.
    The maximum size of the TINYINT and REAL data types is changed to correctly reflect the precision of the respective data types.
    The following table shows the maximum lengths of the data types now and in previous versions:

    ```
                        V5.1 and    Versions previous
    Data type           higher      to V5.1
    ----------          --------    -----------------
    TINYINT              4           6
    REAL                15          24
    ```
- The Guide to Using Oracle SQL/Services Client APIs does not describe that the sqlsrv_associate() service returns SQL error code −1028 when connecting to a database service if the user has not been granted the right to attach to the database.
  When a user connects to a database service, the sqlsrv_associate() service completes with the SQL error code −1028, SQL_NO_PRIV, if the user has been granted access to the Oracle SQL/Services service, but has not been granted the right to attach to the database. A record of the failure is written to the executor process's log file. Note that the sqlsrv_associate() service completes with the Oracle SQL/Services error code −2034, SQLSRV_GETACCINF if the user has not been granted access to the Oracle SQL/Services service.

# Chapter 9
# Known Problems and Restrictions

This chapter describes problems and restrictions relating to Oracle Rdb Release 7.1.2.4, and includes workarounds where appropriate.

# 9.1 Known Problems and Restrictions in All Interfaces

This section describes known problems and restrictions that affect all interfaces for Release 7.1.

## 9.1.1 New Attributes Saved by RMU/LOAD Incompatible With Prior Versions

Bug 2676851

To improve the behavior of unloading views, Oracle Rdb Release 7.1.2 changed the way view columns were unloaded so that attributes for view computed columns, COMPUTED BY and AUTOMATIC columns were saved. These new attributes are not accepted by prior releases of Oracle Rdb.

The following example shows the reported error trying to load a file from V7.1.2 under V7.1.0.4.

```
%RMU−F−NOTUNLFIL, Input file was not created by RMU UNLOAD
%RMU−I−DATRECSTO,   0 data records stored.
%RMU−F−FTL_LOAD, Fatal error for LOAD operation at 21−OCT−2003 16:34:54.20
```

You can workaround this problem by using the /RECORD_DEFINITION qualifier and specifying the FORMAT=DELIMITED option. However, this technique does not support LIST OF BYTE VARYING column unloading.

## 9.1.2 SYSTEM−F−INSFMEM Fatal Error With SHARED MEMORY IS SYSTEM or LARGE MEMORY IS ENABLED in Galaxy Environment

When using the GALAXY SUPPORT IS ENABLED feature in an OpenVMS Galaxy environment, a *%SYSTEM−F−INSFMEM, insufficient dynamic memory error* may be returned when mapping record caches or opening the database. One source of this problem specific to a Galaxy configuration is running out of Galaxy Shared Memory regions. For Galaxy systems, GLX_SHM_REG is the number of shared memory region structures configured into the Galaxy Management Database (GMDB).

While the default value (for OpenVMS versions through at least V7.3−1) of 64 regions might be adequate for some installations, sites using a larger number of databases or row caches when the SHARED MEMORY IS SYSTEM or LARGE MEMORY IS ENABLED features are enabled may find the default insufficient.

If a *%SYSTEM−F−INSFMEM, insufficient dynamic memory* error is returned when mapping record caches or opening databases, Oracle Corporation recommends that you increase the GLX_SHM_REG parameter by 2 times the sum of the number of row caches and number of databases that might be accessed in the Galaxy at one time. As the Galaxy shared memory region structures are not very large, setting this parameter to a higher than required value does not consume a significant amount of physical memory. It also may avoid a later reboot of the Galaxy environment. This parameter must be set on all nodes in the Galaxy.

Galaxy Reboot Required

***Changing the GLX_SHM_REG system parameter requires that the OpenVMS Galaxy environment be booted from scratch. That is, all nodes in the Galaxy must be shut down and then the Galaxy reformed by starting each instance.***

## 9.1.3 Oracle Rdb and OpenVMS ODS−5 Volumes

The OpenVMS Version 7.2 release introduced an Extended File Specifications feature, which consists of two major components:

- A new, optional, volume structure, ODS−5, which provides support for file names that are longer and have a greater range of legal characters than in previous versions of OpenVMS.
- Support for "deep" directory trees.

ODS−5 was introduced primarily to provide enhanced file sharing capabilities for users of Advanced Server for OpenVMS 7.2 (formerly known as PATHWORKS for OpenVMS), as well as DCOM and JAVA applications.

In some cases, Oracle Rdb performs its own file and directory name parsing and explicitly requires ODS−2 (the traditional OpenVMS volume structure) file and directory name conventions to be followed. Because of this knowledge, Oracle does not support any Oracle Rdb database file components (including root files, storage area files, after image journal files, record cache backing store files, database backup files, after image journal backup files, etc.) that utilize any non−ODS−2 file naming features. For this reason, Oracle recommends that Oracle Rdb database components not be located on ODS−5 volumes.

Oracle does support Oracle Rdb database file components on ODS−5 volumes provided that all of these files and directories used by Oracle Rdb strictly follow the ODS−2 file and directory name conventions. In particular, all file names must be specified entirely in uppercase and "special" characters in file or directory names are forbidden.

## 9.1.4 Optimization of Check Constraints

Bug 1448422

When phrasing constraints using the "CHECK" syntax, a poorer strategy can be chosen by the optimizer than when the same or similar constraint is phrased using referential integrity (PRIMARY and FOREIGN KEY) constraints.

For example, I have two tables T1 and T2, both with one column, and I wish to ensure that all values in table T1 exist in T2. Both tables have an index on the referenced field. I could use a PRIMARY KEY constraint on T2 and a FOREIGN KEY constraint on T1.

```
SQL> alter table t2
cont>   alter column f2 primary key not deferrable;
SQL> alter table t1
cont>   alter column f1 references t2 not deferrable;
```

When deleting from the PRIMARY KEY table, Rdb will only check for rows in the FOREIGN KEY table where the FOREIGN KEY has the deleted value. This can be seen as an index lookup on T1 in the retrieval strategy.

```
SQL> delete from t2 where f2=1;
Get     Temporary relation      Retrieval by index of relation T2
  Index name  I2 [1:1]
Index only retrieval of relation T1
  Index name  I1 [1:1]
%RDB−E−INTEG_FAIL, violation of constraint T1_FOREIGN1 caused operation to fail
```

The failure of the constraint is not important. What is important is that Rdb efficiently detects that only those rows in T1 with the same values as the deleted row in T2 can be affected.

It is necessary sometimes to define this type of relationship using CHECK constraints. This could be necessary because the presence of NULL values in the table T2 precludes the definition of a primary key on that table. This could be done with a CHECK constraint of the form:

```
SQL> alter table t1
cont>   alter column f1
cont>   check (f1 in (select * from t2)) not deferrable;
SQL> delete from t2 where f2=1;
Get     Temporary relation      Retrieval by index of relation T2
  Index name  I2 [1:1]
Cross block of 2 entries
  Cross block entry 1
    Index only retrieval of relation T1
      Index name  I1 [0:0]
  Cross block entry 2
    Conjunct        Aggregate−F1    Conjunct
    Index only retrieval of relation T2
      Index name  I2 [0:0]
%RDB−E−INTEG_FAIL, violation of constraint T1_CHECK1 caused operation to fail
```

The cross block is for the constraint evaluation. This retrieval strategy indicates that to evaluate the constraint, the entire index on table T1 is being scanned and for each key, the entire index in table T2 is being scanned. The behavior can be improved somewhat by using an equality join condition in the select clause of the constraint:

```
SQL> alter table t1
cont>   alter column f1
cont>   check (f1 in (select * from t2 where f2=f1))
cont>      not deferrable;

or:

SQL> alter table t1
cont>   alter column f1
cont>   check (f1=(select * from t2 where f2=f1))
cont>      not deferrable;
```

In both cases the retrieval strategy will look like this:

```
SQL> delete from t2 where f2=1;
Get     Temporary relation      Retrieval by index of relation T2
  Index name  I2 [1:1]
Cross block of 2 entries
```

```
  Cross block entry 1
    Index only retrieval of relation T1
      Index name  I1 [0:0]
  Cross block entry 2
    Conjunct        Aggregate−F1    Conjunct
    Index only retrieval of relation T2
      Index name  I2 [1:1]
%RDB−E−INTEG_FAIL, violation of constraint T1_CHECK1 caused operation to fail
```

While the entire T1 index is scanned, at least the value from T1 is used to perform an index lookup on T2.

These restrictions result from semantic differences in the behavior of the "IN" and "EXISTS" operators with respect to null handling, and the complexity of dealing with non−equality join conditions.

To improve the performance of this type of integrity check on larger tables, it is possible to use a series of triggers to perform the constraint check. The following triggers perform a similar check to the constraints above.

```
SQL> create trigger t1_insert
cont>  after insert on t1
cont>  when (not exists (select * from t2 where f2=f1))
cont>    (error) for each row;
SQL> create trigger t1_update
cont>  after update on t1
cont>  when (not exists (select * from t2 where f2=f1))
cont>    (error) for each row;
SQL> ! A delete trigger is not needed on T1.
SQL> create trigger t2_delete
cont>  before delete on t2
cont>  when (exists (select * from t1 where f1=f2))
cont>    (error) for each row;
SQL> create trigger t2_modify
cont>  after update on t2
cont>  referencing old as t2o new as t2n
cont>  when (exists (select * from t1 where f1=t2o.f2))
cont>    (error) for each row;
SQL> ! An insert trigger is not needed on T2.
```

The strategy for a delete on T2 is now:

```
SQL> delete from t2 where f2=1;
Aggregate−F1    Index only retrieval of relation T1
  Index name  I1 [1:1]
Temporary relation      Get     Retrieval by index of relation T2
  Index name  I2 [1:1]
%RDB−E−TRIG_INV_UPD, invalid update; encountered error condition defined for
trigger
−RDMS−E−TRIG_ERROR, trigger T2_DELETE forced an error
```

The trigger strategy is the index only retrieval displayed first. You will note that the index on T1 is used to examine only those rows that may be affected by the delete.

Care must be taken when using this workaround as there are semantic differences in the operation of the triggers, the use of "IN" and "EXISTS", and the use of referential integrity constraints.

This workaround is useful where the form of the constraint is more complex, and cannot be phrased using referential integrity constraints. For example, if the application is such that the value in table T1 may be spaces or NULL to indicate the absence of a value, the above triggers could easily be modified to allow for these semantics.

## 9.1.5 Using Databases from Releases Earlier Than V6.0

You cannot convert or restore databases earlier than V6.0 directly to V7.1. The RMU Convert command for V7.1 supports conversions from V6.0 through V7.0 only. If you have a V3.0 through V5.1 database, you must convert it to at least V6.0 and then convert it to V7.1. For example, if you have a V4.2 database, convert it first to at least V6.0, then convert the resulting database to V7.1.

If you attempt to convert a database created prior to V6.0 directly to V7.1, Oracle RMU generates an error.

## 9.1.6 Carryover Locks and NOWAIT Transaction Clarification

In NOWAIT transactions, the BLAST (Blocking AST) mechanism cannot be used. For the blocking user to receive the BLAST signal, the requesting user must request the locked resource with WAIT (which a NOWAIT transaction does not do). Oracle Rdb defines a resource called NOWAIT, which is used to indicate that a NOWAIT transaction has been started. When a NOWAIT transaction starts, the user requests the NOWAIT resource. All other database users hold a lock on the NOWAIT resource so that when the NOWAIT transaction starts, all other users are notified with a NOWAIT BLAST. The BLAST causes blocking users to release any carryover locks. There can be a delay before the transactions with carryover locks detect the presence of the NOWAIT transaction and release their carryover locks. You can detect this condition by examining the stall messages. If the "Waiting for NOWAIT signal (CW)" stall message appears frequently, the application is probably experiencing a decrease in performance, and you should consider disabling the carryover lock behavior.

## 9.1.7 Unexpected Results Occur During Read−Only Transactions on a Hot Standby Database

When using Hot Standby, it is typical to use the standby database for reporting, simple queries, and other read−only transactions. If you are performing these types of read−only transactions on a standby database, be sure you can tolerate a READ COMMIT level of isolation. This is because the Hot Standby database might be updated by another transaction before the read−only transaction finishes, and the data retrieved might not be what you expected.

Because Hot Standby does not write to the snapshot files, the isolation level achieved on the standby database for any read−only transaction is a READ COMMITED transaction. This means that nonrepeatable reads and phantom reads are allowed during the read−only transaction:

- Nonrepeatable read operations: Allows the return of different results within a single transaction when an SQL operation reads the same row in a table twice. Nonrepeatable reads can occur when another transaction modifies and commits a change to the row between transactions. Because the standby database will update the data when it confirms a transaction has been committed, it is very possible to see an SQL operation on a standby database return different results.

- Phantom read operations: Allows the return of different results within a single transaction when an SQL operation retrieves a range of data values (or similar data existence check) twice. Phantoms can occur if another transaction inserted a new record and committed the insertion between executions of the range retrieval. Again, because the standby database may do this, phantom reads are possible.

Thus, you cannot rely on any data read from the standby database to remain unchanged. Be sure your read−only transactions can tolerate a READ COMMIT level of isolation before you implement procedures that read and use data from a standby database.

## 9.1.8 Both Application and Oracle Rdb Using SYS$HIBER

In application processes that use Oracle Rdb and the $HIBER system service (possibly through RTL routines such as LIB$WAIT), the application must ensure that the event being waited for has actually occurred. Oracle Rdb uses $HIBER/$WAKE sequences for interprocess communications particularly when the ALS (AIJ Log Server) feature is enabled.

The use of the $WAKE system service by Oracle Rdb can interfere with other users of $HIBER (such as the routine LIB$WAIT) that do not check for event completion, possibly causing a $HIBER to be unexpectedly resumed without waiting at all.

To avoid these situations, consider altering the application to use a code sequence that avoids continuing without a check for the operation (such as a delay or a timer firing) being complete.

The following pseudo−code shows how a flag can be used to indicate that a timed−wait has completed correctly. The wait does not complete until the timer has actually fired and set TIMER_FLAG to TRUE. This code relies on ASTs being enabled.

```
ROUTINE TIMER_WAIT:
    BEGIN
    ! Clear the timer flag
    TIMER_FLAG = FALSE
    ! Schedule an AST for sometime in the future
    STAT = SYS$SETIMR (TIMADR = DELTATIME, ASTRTN = TIMER_AST)
    IF STAT <> SS$_NORMAL
    THEN BEGIN
         LIB$SIGNAL (STAT)
         END
    ! Hibernate.  When the $HIBER completes, check to make
    ! sure that TIMER_FLAG is set indicating that the wait
    ! has finished.
    WHILE TIMER_FLAG = FALSE
    DO  BEGIN
        SYS$HIBER()
        END
    END
ROUTINE TIMER_AST:
    BEGIN
    ! Set the flag indicating that the timer has expired
    TIMER_FLAG = TRUE
    ! Wake the main−line code
    STAT = SYS$WAKE ()
    IF STAT <> SS$_NORMAL
    THEN BEGIN
         LIB$SIGNAL (STAT)
         END
```

```
END
```

The LIB$K_NOWAKE flag can be specified when using the OpenVMS LIB$WAIT routine to allow an alternate wait scheme (using the $SYNCH system service) that can avoid potential problems with multiple code sequences using the $HIBER system service.

## 9.1.9 Bugcheck Dump Files with Exceptions at COSI_CHF_SIGNAL

In certain situations, Oracle Rdb bugcheck dump files indicate an exception at COSI_CHF_SIGNAL. This location is, however, not the address of the actual exception. The actual exception occurred at the previous call frame on the stack (the one listed as the next Saved PC after the exception).

For example, consider the following bugcheck file stack information:

```
$ SEARCH RDSBUGCHK.DMP "EXCEPTION","SAVED PC","−F−","−E−"

***** Exception at 00EFA828 : COSI_CHF_SIGNAL + 00000140
%COSI−F−BUGCHECK, internal consistency failure
Saved PC = 00C386F0 : PSIINDEX2JOINSCR + 00000318
Saved PC = 00C0BE6C : PSII2BALANCE + 0000105C
Saved PC = 00C0F4D4 : PSII2INSERTT + 000005CC
Saved PC = 00C10640 : PSII2INSERTTREE + 000001A0
      .
      .
      .
```

In this example, the exception actually occurred at PSIINDEX2JOINSCR offset 00000318. If you have a bugcheck dump with an exception at COSI_CHF_SIGNAL, it is important to note the next "Saved PC" because it is needed when working with Oracle Rdb Worldwide Support.

## 9.1.10 Read−only Transactions Fetch AIP Pages Too Often

Oracle Rdb read−only transactions fetch Area Inventory Pages (AIP) to ensure that the logical area has not been modified by an exclusive read−write transaction. This check is needed because an exclusive read−write transaction does not write snapshot pages and these pages may be needed by the read−only transaction.

Because AIPs are always stored in the RDB$SYSTEM area, reading the AIP pages could represent a significant amount of I/O to the RDB$SYSTEM area for some applications. Setting the RDB$SYSTEM area to read−only can avoid this problem, but it also prevents other online operations that might be required by the application so it is not a viable workaround in all cases.

This problem has been reduced in Oracle Rdb release 7.0. The AIP entries are now read once and then are not read again unless they need to be. This optimization requires that the carry−over locks feature be enabled (this is the default setting). If carry over locks are not enabled, this optimization is not enabled and the behavior is the same as in previous releases.

## 9.1.11 Row Cache Not Allowed While Hot Standby Replication is Active

The row cache feature may not be enabled on a hot standby database while replication is active. The hot standby feature will not start if row cache is enabled.

This restriction exists because rows in the row cache are accessed via logical dbkeys. However, information transferred to the standby database via the after image journal facility only contains physical dbkeys. Because there is no way to maintain rows in the cache via the hot standby processing, the row cache must be disabled when the standby database is open and replication is active.

A new command qualifier, ROW_CACHE=DISABLED, has been added to the RMU Open command. To open the hot standby database prior to starting replication, use the ROW_CACHE=DISABLED qualifier on the RMU Open command.

## 9.1.12 Excessive Process Page Faults and other Performance Considerations During Oracle Rdb Sorts

Excessive hard or soft page faulting can be a limiting factor of process performance. One factor contributing to Oracle Rdb process page faulting is sorting operations. Common causes of sorts include the SQL GROUP BY, ORDER BY, UNION, and DISTINCT clauses specified for a query, and index creation operations. Defining the logical name RDMS$DEBUG_FLAGS to "RS" can help determine when Oracle Rdb sort operations are occurring and to display the sort keys and statistics.

Oracle Rdb includes its own copy of the OpenVMS SORT32 code within the Oracle Rdb images and does not generally call the routines in the OpenVMS run−time library. A copy of the SORT32 code is used to provide stability between versions of Oracle Rdb and OpenVMS and because Oracle Rdb calls the sort routines from executive processor mode which is difficult to do using the SORT32 shareable image. SQL IMPORT and RMU Load operations do, however, call the OpenVMS SORT run−time library.

At the beginning of a sort operation, the SORT code allocates some memory for working space. The SORT code uses this space for buffers, in−memory copies of the data, and sorting trees.

SORT does not directly consider the processes quotas or parameters when allocating memory. The effects of WSQUOTA and WSEXTENT are indirect. At the beginning of each sort operation, the SORT code attempts to adjust the process working set to the maximum possible size using the $ADJWSL system service specifying a requested working set limit of %X7FFFFFFF pages (the maximum possible). SORT then uses a value of 75% of the returned working set for virtual memory scratch space. The scratch space is then initialized and the sort begins.

The initialization of the scratch space generally causes page faults to access the pages newly added to the working set. Pages that were in the working set already may be faulted out as the new pages are faulted in. Once the sort operation completes and SORT returns back to Oracle Rdb, the pages that may have been faulted out of the working set are likely to be faulted back into the working set.

When a process working set is limited by the working set quota (WSQUOTA) parameter and the working set extent (WSEXTENT) parameter is a much larger value, the first call to the sort routines can cause many page faults as the working set grows. Using a value of WSEXTENT that is closer to WSQUOTA can help reduce the impact of this case.

With some OpenVMS versions, AUTOGEN sets the SYSGEN parameter PQL_MWSEXTENT equal to the WSMAX parameter. This means that all processes on the system end up with WSEXTENT the same as WSMAX. Since that might be quite high, sorting might result in excessive page faulting. You may want to explicitly set PQL_MWSEXTENT to a lower value if this is the case on your system.

Sort work files are another factor to consider when tuning for Oracle Rdb sort operations. When the operation can not be done in the available memory, SORT uses temporary disk files to hold the data as it is being sorted. The Oracle Rdb7 Guide to Database Performance and Tuning contains more detailed information about sort work files.

The logical name RDMS$BIND_SORT_WORKFILES specifies how many work files sort is to use if work files are required. The default is 2 and the maximum number is 10. The work files can be individually controlled by the SORTWORKn logical names (where n is from 0 through 9). You can increase the efficiency of sort operations by assigning the location of the temporary sort work files to different disks. These assignments are made by using up to ten logical names, SORTWORK0 through SORTWORK9.

Normally, SORT places work files in the your SYS$SCRATCH directory. By default, SYS$SCRATCH is the same device and directory as the SYS$LOGIN location. Spreading the I/O load over many disks improves efficiency as well as performance by taking advantage of the system resources and helps prevent disk I/O bottlenecks. Specifying that a your work files reside on separate disks permits overlap of the SORT read/write cycle. You may also encounter cases where insufficient space exists on the SYS$SCRATCH disk device (for example, while Oracle Rdb builds indexes for a very large table). Using the SORTWORK0 through SORTWORK9 logical names can help you avoid this problem.

Note that SORT uses the work files for different sorted runs, and then merges the sorted runs into larger groups. If the source data is mostly sorted, then not every sort work file may need to be accessed. This is a possible source of confusion because even with 10 sort work files, it is possible to exceed the capacity of the first SORT file and the sort operation fails never having accessed the remaining 9 sort work files.

Note that the logical names RDMS$BIND_WORK_VM and RDMS$BIND_WORK_FILE do not affect or control the operation of sort. These logical names are used to control other temporary space allocation within Oracle Rdb.

## 9.1.13 Control of Sort Work Memory Allocation

Oracle Rdb uses a built−in SORT32 package to perform many sort operations. Sometimes, these sorts exhibit a significant performance problem when initializing work memory to be used for the sort. This behavior can be experienced, for example, when a very large sort cardinality is estimated, but the actual sort cardinality is small.

In rare cases, it may be desirable to artificially limit the sort package's use of work memory. Two logicals have been created to allow this control. In general, there should be no need to use either of these logicals and misuse of them can significantly impact sort performance. Oracle recommends that these logicals be used carefully and sparingly.

The logical names are:

***Table 9−1 Sort Memory Logicals***

| Logical | Definition |
|---|---|
| RDMS$BIND_SORT_MEMORY_WS_FACTOR | Specifies a percentage of the process's working set limit to be used when allocating sort memory for the built−in SORT32 package. If not defined, the default value is 75 (representing 75%), the maximum value is 75 (representing 75%), and the minimum value is 2 (representing 2%). Processes with vary large working set limits can sometimes experience significant page faulting and CPU consumption while initializing sort memory. This logical name can restrict the sort work memory to a percentage of the processes maximum working set. |
| RDMS$BIND_SORT_MEMORY_MAX_BYTES | Specifies an absolute limit to be used when allocating sort memory for the built−in SORT32 package. If not defined, the default value is unlimited (up to 1GB), the maximum value is 2,147,483,647 and the minimum value is 32,768. |

## 9.1.14 The Halloween Problem

When a cursor is processing rows selected from a table, it is possible that another separate query can interfere with the retrieval of the cursor by modifying the index columns key values used by the cursor.

For instance, if a cursor selects all EMPLOYEES with LAST_NAME >= 'M', it is likely that the query will use the sorted index on LAST_NAME to retrieve the rows for the cursor. If an update occurs during the processing of the cursor which changes the LAST_NAME of an employee from "Mason" to "Rickard", then it is possible that that employee row will be processed twice. First when it is fetched with name "Mason", and then later when it is accessed by the new name "Rickard".

The Halloween problem is a well known problem in relational databases. Access strategies which optimize the I/O requirements, such as Index Retrieval, can be subject to this problem. Interference from queries by other sessions are avoided by locking and are controlled by the ISOLATION LEVEL options in SQL, or the CONCURRENCY/CONSISTENCY options in RDO/RDML.

Oracle Rdb avoids this problem if it knows that the cursors subject table will be updated. For example, if the SQL syntax UPDATE ... WHERE CURRENT OF is used to perform updates of target rows, or the RDO/RDML MODIFY statement uses the context variable for the stream. Then the optimizer will choose an alternate access strategy if an update can occur which may cause the Halloween problem. This can be seen in the access strategy in Example 2−2 as a "Temporary relation" being created to hold the result of the cursor query.

When you use interactive or dynamic SQL, the UPDATE ... WHERE CURRENT OF or DELETE ... WHERE CURRENT OF statements will not be seen until after the cursor is declared and opened. In these environments, you must use the FOR UPDATE clause to specify that columns selected by the cursor will be updated during cursor processing. This is an indication to the Rdb optimizer so that it protects against the Halloween problem in this case. This is shown in Example 2−1 and Example 2−2.

The following example shows that the EMP_LAST_NAME index is used for retrieval. Any update performed will possibly be subject to the Halloween problem.

```
SQL> set flags 'strategy';
SQL> declare emp  cursor for
cont> select * from employees where last_name >= 'M'
cont> order by last_name;
SQL> open emp;
Conjunct        Get     Retrieval by index of relation EMPLOYEES
  Index name  EMP_LAST_NAME [1:0]
SQL> close emp;
```

The following example shows that the query specifies that the column LAST_NAME will be updated by some later query. Now the optimizer protects the EMP_LAST_NAME index used for retrieval by using a "Temporary Relation" to hold the query result set. Any update performed on LAST_NAME will now avoid the Halloween problem.

```
SQL> set flags 'strategy';
SQL> declare emp2 cursor for
cont> select * from employees where last_name >= 'M'
cont> order by last_name
cont> for update of last_name;
SQL> open emp2;
Temporary relation      Conjunct        Get
Retrieval by index of relation EMPLOYEES
  Index name  EMP_LAST_NAME [1:0]
SQL> close emp2;
```

When you use the SQL precompiler, or the SQL module language compiler it can be determined from usage that the cursor context will possibly be updated during the processing of the cursor because all cursor related statements are present within the module. This is also true for the RDML/RDBPRE precompilers when you use the DECLARE_STREAM and START_STREAM statements and use the same stream context to perform all MODIFY and ERASE statements.

The point to note here is that the protection takes place during the open of the SQL cursor (or RDO stream), not during the subsequent UPDATE or DELETE.

If you execute a separate UPDATE query which modifies rows being fetched from the cursor then the actual rows fetched will depend upon the access strategy chosen by the Rdb optimizer. As the query is separate from the cursors query (i.e. doesn't reference the cursor context), then the optimizer does not know that the cursor selected rows are potentially updated and so cannot perform the normal protection against the Halloween problem.

# 9.2 SQL Known Problems and Restrictions

This section describes known problems and restrictions for the SQL interface for release 7.1.

## 9.2.1 SET FLAGS CRONO_FLAG to be Removed

The SET FLAGS statement and RDMS$SET_FLAGS logical name currently accept the obsolete keyword CRONO_FLAG. This keyword will be removed in the next release of Oracle Rdb V7.1. Please update all scripts and applications to use the keyword CHRONO_FLAG.

## 9.2.2 Interchange File (RBR) Created by Oracle Rdb Release 7.1 Not Compatible With Previous Releases

To support the large number of new database attributes and objects, the protocol used by SQL EXPORT and SQL IMPORT has been enhanced to support more protocol types. Therefore, this format of the Oracle Rdb release 7.1 interchange files can no longer be read by older versions of Oracle Rdb.

Oracle Rdb continues to provide upward compatibility for interchange files generated by older versions.

Oracle Rdb has never supported backward compatibility, however, it was sometimes possible to use an interchange file with an older version of IMPORT. However, this protocol change will no longer permit this usage.

## 9.2.3 Unexpected NO_META_UPDATE Error Generated by DROP MODULE ... CASCADE When Attached by PATHNAME

The SQL DROP MODULE ... CASCADE statement may sometimes generate an unexpected NO_META_UPDATE error. This occurs when the session attaches to a database by PATHNAME. For example:

```
SQL> drop module m1 cascade;
%RDB−E−NO_META_UPDATE, metadata update failed
−RDMS−F−OBJ_INUSE, object "M1P1" is referenced by M2.M2P1 (usage: Procedure)
−RDMS−E−MODNOTDEL, module "M1" has not been deleted
```

This error occurs because the CASCADE option is ignored because the Oracle CDD/Repository does not support CASCADE. The workaround is to attach by FILENAME and perform the metadata operation.

In a future release of Oracle Rdb, an informational message will be issued describing the downgrade from CASCADE to RESTRICT in such cases.

## 9.2.4 System Relation Change for International Database Users

Due to an error in creating the RDB$FIELD_VERSIONS system relation, another system relation, RDB$STORAGE_MAP_AREAS, cannot be accessed if the session character sets are not set to DEC_MCS.

This problem prevents the new Oracle Rdb GUIs, specifically the Oracle Rdb Schema Manager, from viewing indexes and storage maps from existing Oracle Rdb databases.

The problem can be easily corrected by executing the following SQL statement after attaching to the database:

```
SQL> UPDATE RDB$FIELD_VERSIONS SET RDB$FIELD_SUB_TYPE = 32767
cont> WHERE RDB$FIELD_NAME = 'RDB$AREA_NAME';
```

## 9.2.5 Single Statement LOCK TABLE is Not Supported for SQL Module Language and SQL Precompiler

The new LOCK TABLE statement is not currently supported as a single statement within the module language or embedded SQL language compiler.

Instead you must enclose the statement in a compound statement. That is, use BEGIN... END around the statement as shown in the following example. This format provides all the syntax and flexibility of LOCK TABLE.

This restriction does not apply to interactive or dynamic SQL.

The following extract from the module language listing file shows the reported error if you use LOCK TABLE as a single statement procedure. The other procedure in the same module is acceptable because it uses a compound statement that contains the LOCK TABLE statement.

```
MODULE sample_test
LANGUAGE C
PARAMETER COLONS

DECLARE ALIAS FILENAME 'mf_personnel'

PROCEDURE a (SQLCODE);
LOCK TABLE employees FOR EXCLUSIVE WRITE MODE;
%SQL−F−WISH_LIST, (1) Feature not yet implemented − LOCK TABLE requires compound
statement

PROCEDURE b (SQLCODE);
BEGIN
LOCK TABLE employees FOR EXCLUSIVE WRITE MODE;
END;
```

To workaround this problem of using LOCK TABLE for SQL module language or embedded SQL application, use a compound statement in an EXEC SQL statement.

## 9.2.6 Multistatement or Stored Procedures May Cause Hangs

Long−running multistatement or stored procedures can cause other users in the database to hang if the procedures obtain resources needed by those other users. Some resources obtained by the execution of a multistatement or stored procedure are not released until the multistatement or stored procedure finishes. Thus, any−long running multistatement or stored procedure can cause other processes to hang. This problem can be encountered even if the statement contains SQL COMMIT or ROLLBACK statements.

The following example demonstrates the problem. The first session enters an endless loop; the second session attempts to backup the database but hangs forever.

```
Session 1:

SQL> attach 'filename MF_PERSONNEL';
SQL> create function LIB$WAIT (in real by reference)
cont> returns integer;
cont> external name LIB$WAIT location 'SYS$SHARE:LIBRTL.EXE'
cont> language general general parameter style variant;
SQL> commit;
        .
        .
        .
$ SQL
SQL> attach 'filename MF_PERSONNEL';
SQL> begin
cont> declare :LAST_NAME LAST_NAME_DOM;
cont> declare :WAIT_STATUS integer;
cont> loop
cont> select LAST_NAME into :LAST_NAME
cont> from EMPLOYEES where EMPLOYEE_ID = '00164';
cont> rollback;
cont> set :WAIT_STATUS = LIBWAIT (5.0);
cont> set transaction read only;
cont> end loop;
cont> end;

Session 2:

$ RMU/BACKUP/LOG/ONLINE MF_PERSONNEL MF_PERSONNEL

From a third session, you can see that the backup process is waiting
for a lock held in the first session:

$ RMU/SHOW LOCKS /MODE=BLOCKING MF_PERSONNEL
    .
    .
    .
Resource: nowait signal

ProcessID Process Name       Lock ID  System ID Requested Granted
--------- ---------------    -------- --------- --------- -------
20204383  RMU BACKUP.....    5600A476  00010001 CW        NL
2020437B  SQL............    3B00A35C  00010001 PR        PR
```

There is no workaround for this restriction. When the multistatement or stored procedure finishes execution, the resources needed by other processes are released.

## 9.2.7 Use of Oracle Rdb from Shareable Images

If code in the image initialization routine of a shareable image makes any calls into Oracle Rdb, through SQL or any other means, access violations or other unexpected behavior may occur if Oracle Rdb images have not had a chance to do their own initialization.

To avoid this problem, applications must take one of the following steps:

- Do not make Oracle Rdb calls from the initialization routines of shareable images.
- Link in such a way that the RDBSHR.EXE image initializes first. You can do this by placing the reference to RDBSHR.EXE and any other Oracle Rdb shareable images last in the linker options file.

This is not a bug; it is a restriction resulting from the way OpenVMS image activation works.

# 9.3 Oracle RMU Known Problems and Restrictions

This section describes known problems and restrictions for the RMU interface for release 7.1.

## 9.3.1 RMU/BACKUP MAX_FILE_SIZE Option Has Been Disabled

The MAX_FILE_SIZE option of the RMU/BACKUP/DISK_FILE qualifier for backup to multiple disk files has been temporarily disabled since it creates corrupt RBF files if the maximum file size in megabytes is exceeded and a new RBF file is created. It also does not give a unique name to the new RBF file but creates an RBF file with the same name but a new version number in the same disk directory. This will cause an RMU−F−BACFILCOR error on the restore and the restore will not complete.

The multi−file disk backup and restore will succeed if this option is not used. If this option is specified, a warning message is now output that this qualifier will be ignored.

The following example shows that the MAX_FILE_SIZE option, when used with the /DISK_FILE qualifier on an RMU/BACKUP, will be ignored and a warning message will be output.

```
$ RMU/BACKUP /ONLINE                    -
             /NOCRC                     -
             /NOLOG                     -
             /NOINCREMENTAL             -
             /QUIET_POINT               -
             TEST_DB_DIR:TEST_DB
  −
BACKUP_DIR_1:TEST_DB/DISK_FILE=(WRITER_THREADS=3,MAX_FILE_SIZE=10) ,−
BACKUP_DIR_2:/DISK_FILE=(WRITER_THREADS=3,MAX_FILE_SIZE=10) ,−
BACKUP_DIR_3:/DISK_FILE=(WRITER_THREADS=3,MAX_FILE_SIZE=10)

%RMU−W−DISABLEDOPTION, The MAX_FILE_SIZE option is temporarily disabled
        and will be ignored
```

As a workaround to avoid this problem, do not specify the MAX_FILE_SIZE option with the /DISK_FILE qualifier.

## 9.3.2 RMU Convert Fails When Maximum Relation ID is Exceeded

If, when relation IDs are assigned to new system tables during an RMU Convert of an Oracle Rdb V7.0 database to a V7.1 database, the maximum relation ID of 8192 allowed by Oracle Rdb is exceeded, the fatal error %RMU−F−RELMAXIDBAD is displayed and the database is rolled back to V70. Contact your Oracle support representative if you get this error. Note that when the database is rolled back, the fatal error %RMU−F−CVTROLSUC is displayed to indicate that the rollback was successful but caused by the detection of a fatal error and not requested by the user.

This condition only occurs if there are an extremely large number of tables defined in the database or if a large number of tables were defined but have subsequently been deleted.

The following example shows both the %RMU−F−RELMAXIDBAD error message if the allowed database relation ID maximum of 8192 is exceeded and the %RMU−F−CVTROLSUC error message when the database has been rolled back to V7.0 since it cannot be converted to V7.1:

```
$rmu/convert mf_personnel
%RMU−I−RMUTXT_000, Executing RMU for Oracle Rdb V7.1−00
Are you satisfied with your backup of
 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 and your backup of
 any associated .aij files [N]? Y
 %RMU−I−LOGCONVRT, database root converted to current structure level
 %RMU−F−RELMAXIDBAD, ROLLING BACK CONVERSION − Relation ID exceeds maximum
  8192 for system table RDB$RELATIONS
 %RMU−F−CVTROLSUC, CONVERT rolled−back for
  DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 to version V7.0
```

The following example shows the normal case when the maximum allowed relation ID is not exceeded:

```
$rmu/convert mf_personnel
%RMU−I−RMUTXT_000, Executing RMU for Oracle Rdb V7.1−00
Are you satisfied with your backup of
 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 and your backup of
 any associated .aij files [N]? Y
%RMU−I−LOGCONVRT, database root converted to current structure level
%RMU−S−CVTDBSUC, database DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1
 successfully converted from version V7.0 to V7.1
%RMU−I−CVTCOMSUC, CONVERT committed for
 DEVICE:[DIRECTORY]MF_PERSONNEL.RDB;1 to version V7.1
```

## 9.3.3 RMU Unload /After_Journal Requires Accurate AIP Logical Area Information

The RMU Unload /After_Journal command uses the on−disk area inventory pages (AIPs) to determine the appropriate type of each logical area when reconstructing logical dbkeys for records stored in mixed−format storage areas. However, the logical area type information in the AIP is generally unknown for logical areas created prior to Oracle Rdb release 7.0.1. If the RMU Unload /After_Journal command cannot determine the logical area type for one or more AIP entries, a warning message is displayed for each such area and may ultimately return logical dbkeys with a 0 (zero) area number for records stored in mixed−format storage areas.

In order to update the on−disk logical area type in the AIP, the RMU Repair utility must be used. The INITIALIZE=LAREA_PARAMETERS=optionfile qualifier option file can be used with the TYPE qualifier. For example, to repair the EMPLOYEES table of the MF_PERSONNEL database, you would create an options file that contains the following line:

EMPLOYEES /TYPE=TABLE

For partitioned logical areas, the AREA=name qualifier can be used to identify the specific storage areas that are to be updated. For example, to repair the EMPLOYEES table of the MF_PERSONNEL database for the EMPID_OVER storage area only, you would create an options file that contains the following line:

EMPLOYEES /AREA=EMPID_OVER /TYPE=TABLE

The TYPE qualifier specifies the type of a logical area. The following keywords are allowed:

- TABLE
  Specifies that the logical area is a data table. This would be a table created using the SQL CREATE TABLE syntax.
- B−TREE
  Specifies that the logical area is a B−tree index. This would be an index created using the SQL CREATE INDEX TYPE IS SORTED syntax.
- HASH
  Specifies that the logical area is a hash index. This would be an index created using the SQL CREATE INDEX TYPE IS HASHED syntax.
- SYSTEM
  Specifies that the logical area is a system record that is used to identify hash buckets. Users cannot explicitly create these types of logical areas.

  ---

  Note

  ***This type should NOT be used for the RDB$SYSTEM logical areas. This type does NOT identify system relations.***

  ---
- BLOB
  Specifies that the logical area is a BLOB repository.

There is no explicit error checking of the type specified for a logical area. However, an incorrect type may cause the RMU Unload /After_Journal command to be unable to correctly return valid, logical dbkeys.

## 9.3.4 Do Not Use HYPERSORT with RMU Optimize After_Journal Command

The OpenVMS Alpha V7.1 operating system introduced the high−performance Sort/Merge utility (also known as HYPERSORT). This utility takes advantage of the OpenVMS Alpha architecture to provide better performance for most sort and merge operations.

The high−performance Sort/Merge utility supports a subset of the SOR routines. Unfortunately, the high−performance Sort/Merge utility does not support several of the interfaces used by the RMU Optimize After_Journal command. In addition, the high−performance Sort/Merge utility reports no error or warning when being called with the unsupported options used by the RMU Optimize After_Journal command.

Because of this, the use of the high−performance Sort/Merge utility is not supported for the RMU Optimize After_Journal command. Do not define the logical name SORTSHR to reference HYPERSORT.EXE.

## 9.3.5 Changes in EXCLUDE and INCLUDE Qualifiers for RMU Backup

The RMU Backup command no longer accepts both the Include and Exclude qualifiers in the same command. This change removes the confusion over exactly what gets backed up when Include and Exclude are specified on the same line, but does not diminish the capabilities of the RMU Backup command.

To explicitly exclude some storage areas from a backup, use the Exclude qualifier to name the storage areas to be excluded. This causes all storage areas to be backed up except for those named by the Exclude qualifier.

Similarly, the Include qualifier causes only those storage areas named by the qualifier to be backed up. Any storage area not named by the Include qualifier is not backed up. The Noread_only and Noworm qualifiers continue to cause read−only storage areas and WORM storage areas to be omitted from the backup even if these areas are explicitly listed by the Include qualifier.

Another related change is in the behavior of EXCLUDE=*. In previous versions, EXCLUDE=* caused all storage areas to be backed up. Beginning with V7.1, EXCLUDE=* causes only a root backup to be done. A backup created by using EXCLUDE=* can be used only by the RMU Restore Only_Root command.

## 9.3.6 RMU Backup Operations Should Use Only One Type of Tape Drive

When using more than one tape drive for an RMU Backup command, all of the tape drives must be of the same type (for example, all the tape drives must be TA90s or TZ87s or TK50s). Using different tape drive types (for example, one TK50 and one TA90) for a single database backup operation may make database restoration difficult or impossible.

Oracle RMU attempts to prevent using different tape drive densities during a backup operation, but is not able to detect all invalid cases and expects that all tape drives for a backup are of the same type.

As long as all of the tapes used during a backup operation can be read by the same type of tape drive during a restore operation, the backup is likely valid. This may be the case, for example, when using a TA90 and a TA90E.

Oracle Corporation recommends that, on a regular basis, you test your backup and recovery procedures and environment using a test system. You should restore the database and then recover using AIJs to simulate failure recovery of the production system.

Consult the Oracle Rdb7 Guide to Database Maintenance, the Oracle Rdb7 Guide to Database Design and Definition, and the Oracle RMU Reference Manual for additional information about Oracle Rdb backup and restore operations.

## 9.3.7 RMU/VERIFY Reports PGSPAMENT or PGSPMCLST Errors

RMU/VERIFY may sometimes report PGSPAMENT or PGSPMCLST errors when verifying storage areas. These errors indicate that the Space Area Management (SPAM) page fullness threshold for a particular data page does not match the actual space usage on the data page. For a further discussion of SPAM pages, consult the Oracle Rdb7 Guide to Database Maintenance.

In general, these errors will not cause any adverse affect on the operation of the database. There is potential for space on the data page to not be totally utilized, or for a small amount of extra I/O to be expended when searching for space in which to store new rows. But unless there are many of these errors then the impact should be negligible.

It is possible for these inconsistencies to be introduced by errors in Oracle Rdb. When those cases are discovered, Oracle Rdb is corrected to prevent the introduction of the inconsistencies. It is also possible for these errors to be introduced during the normal operation of Oracle Rdb. The following scenario can leave the SPAM pages inconsistent:

1. A process inserts a row on a page, and updates the threshold entry on the corresponding SPAM page to reflect the new space utilization of the data page. The data page and SPAM pages are not flushed to disk.
2. Another process notifies the first process that it would like to access the SPAM page being held by the process. The first process flushes the SPAM page changes to disk and releases the page. Note that it has not flushed the data page.
3. The first process then terminates abnormally (for example, from the DCL STOP/IDENTIFICATION command). Since that process never flushed the data page to disk, it never wrote the changes to the Recovery Unit Journal (RUJ) file. Since there were no changes in the RUJ file for that data page then the Database Recovery (DBR) process did not need to roll back any changes to the page. The SPAM page retains the threshold update change made above even though the data page was never flushed to disk.

While it would be possible to create mechanisms to ensure that SPAM pages do not become out of synch with their corresponding data pages, the performance impact would not be trivial. Since these errors are relatively rare and the impact is not significant, then the introduction of these errors is considered to be part of the normal operation of Oracle Rdb. If it can be proven that the errors are not due to the scenario above, then Oracle Product Support should be contacted.

PGSPAMENT and PGSPMCLST errors may be corrected by doing any one of the following operations:

- Recreate the database by performing:
    1. SQL EXPORT
    2. SQL DROP DATABASE
    3. SQL IMPORT
- Recreate the database by performing:
    1. RMU/BACKUP
    2. SQL DROP DATABASE
    3. RMU/RESTORE
- Repair the SPAM pages by using the RMU/REPAIR command. Note that the RMU/REPAIR command does not write its changes to an after−image journal (AIJ) file. Therefore, Oracle recommends that a full database backup be performed immediately after using the RMU/REPAIR command.

# 9.4 Known Problems and Restrictions in All Interfaces for Release 7.0 and Earlier

The following problems and restrictions from release 7.0 and earlier still exist.

## 9.4.1 Converting Single−File Databases

Because of a substantial increase in the database root file information for V7.0, you should ensure that you have adequate disk space before you use the RMU Convert command with single−file databases and V7.0 or higher.

The size of the database root file of any given database increases a minimum of 13 blocks and a maximum of 597 blocks. The actual increase depends mostly on the maximum number of users specified for the database.

## 9.4.2 Row Caches and Exclusive Access

If a table has a row−level cache defined for it, the Row Cache Server (RCS) may acquire a shared lock on the table and prevent any other user from acquiring a Protective or Exclusive lock on that table.

## 9.4.3 Exclusive Access Transactions May Deadlock with RCS Process

If a table is frequently accessed by long running transactions that request READ/WRITE access reserving the table for EXCLUSIVE WRITE and if the table has one or more indexes, you may experience deadlocks between the user process and the Row Cache Server (RCS) process.

There are at least three suggested workarounds to this problem:

- ◊ Reserve the table for SHARED WRITE
- ◊ Close the database and disable row cache for the duration of the exclusive transaction
- ◊ Change the checkpoint interval for the RCS process to a time longer than the time required to complete the batch job and then trigger a checkpoint just before the batch job starts. Set the interval back to a smaller interval after the checkpoint completes.

## 9.4.4 Strict Partitioning May Scan Extra Partitions

When you use a WHERE clause with the less than (<) or greater than (>) operator and a value that is the same as the boundary value of a storage map, Oracle Rdb scans extra partitions. A boundary value is a value specified in the WITH LIMIT OF clause. The following example, executed while the logical name RDMS$DEBUG_FLAGS is defined as "S", illustrates the behavior:

```
ATTACH 'FILENAME MF_PERSONNEL';
CREATE TABLE T1 (ID INTEGER, LAST_NAME CHAR(12), FIRST_NAME CHAR(12));
CREATE STORAGE MAP M FOR T1 PARTITIONING NOT UPDATABLE
```

```
 STORE USING (ID)
 IN EMPIDS_LOW WITH LIMIT OF (200)
 IN EMPIDS_MID WITH LIMIT OF (400)
 OTHERWISE IN EMPIDS_OVER;
INSERT INTO T1 VALUES (150,'Boney','MaryJean');
INSERT INTO T1 VALUES (350,'Morley','Steven');
INSERT INTO T1 VALUES (300,'Martinez','Nancy');
INSERT INTO T1 VALUES (450,'Gentile','Russ');
SELECT * FROM T1 WHERE ID > 400;
Conjunct Get Retrieval sequentially of relation T1
Strict Partitioning: part 2 3
ID LAST_NAME FIRST_NAME
450 Gentile Russ
1 row selected
```

In the previous example, partition 2 does not need to be scanned. This does not affect the correctness of the result. Users can avoid the extra scan by using values other than the boundary values.

## 9.4.5 Restriction When Adding Storage Areas with Users Attached to Database

If you try to interactively add a new storage area where the page size is less than the existing page size and the database has been manually opened or users are active, the add operation fails with the following error:

```
%RDB−F−SYS_REQUEST, error from system services request
−RDMS−F−FILACCERR, error opening database root DKA0:[RDB]TEST.RDB;1
−SYSTEM−W−ACCONFLICT, file access conflict
```

You can make this change only when no users are attached to the database and, if the database is set to OPEN IS MANUAL, the database is closed. Several internal Oracle Rdb data structures are based on the minimum page size and these structures cannot be resized if users are attached to the database.

Furthermore, because this particular change is not recorded in the AIJ, any recovery scenario fails. Note also that if you use .aij files, you must backup the database and restart after−image journaling because this change invalidates the current AIJ recovery.

## 9.4.6 Support for Single−File Databases to Be Dropped in a Future Release

Oracle Rdb currently supports both single−file and multifile databases on all platforms. However, single−file databases will not be supported in a future release of Oracle Rdb. At that time, Oracle Rdb will provide the means to easily convert single−file databases to multifile databases.

Oracle Rdb recommends that users with single−file databases perform the following actions:

◊ Use the Oracle RMU commands, such as Backup and Restore, to make copies, backup, or move single−file databases. Do not use operating system commands to copy, back up, or move databases.

◊ Create new databases as multifile databases even though single−file databases are supported.

## 9.4.7 Multiblock Page Writes May Require Restore Operation

If a node fails while a multiblock page is being written to disk, the page in the disk becomes inconsistent, and is detected immediately during failover. (Failover is the recovery of an application by restarting it on another computer.) The problem is rare, and occurs because only single−block I/O operations are guaranteed by OpenVMS to be written atomically. This problem has never been reported by any customer and was detected only during stress tests in our labs.

Correct the page by an area−level restore operation. Database integrity is not compromised, but the affected area is not available until the restore operation completes.

A future release of Oracle Rdb will provide a solution that guarantees multiblock atomic write operations. Cluster failovers will automatically cause the recovery of multiblock pages, and no manual intervention will be required.

## 9.4.8 Replication Option Copy Processes Do Not Process Database Pages Ahead of an Application

When a group of copy processes initiated by the Replication Option (formerly Data Distributor) begins running after an application has begun modifying the database, the copy processes catch up to the application and are not able to process database pages that are logically ahead of the application in the RDB$CHANGES system relation. The copy processes all align waiting for the same database page and do not move on until the application has released it. The performance of each copy process degrades because it is being paced by the application.

When a copy process completes updates to its respective remote database, it updates the RDB$TRANSFERS system relation and then tries to delete any RDB$CHANGES rows not needed by any transfers. During this process, the RDB$CHANGES table cannot be updated by any application process, holding up any database updates until the deletion process is complete. The application stalls while waiting for the RDB$CHANGES table. The resulting contention for RDB$CHANGES SPAM pages and data pages severely impacts performance throughput, requiring user intervention with normal processing.

This is a known restriction in V4.0 and higher. Oracle Rdb uses page locks as latches. These latches are held only for the duration of an action on the page and not to the end of transaction. The page locks also have blocking asynchronous system traps (ASTs) associated with them. Therefore, whenever a process requests a page lock, the process holding that page lock is sent a blocking AST (BLAST) by OpenVMS. The process that receives such a blocking AST queues the fact that the page lock should be released as soon as possible. However, the page lock cannot be released immediately.

Such work requests to release page locks are handled at verb commit time. An Oracle Rdb verb is an Oracle Rdb query that executes atomically, within a transaction. Therefore, verbs that require the scan of a large table, for example, can be quite long. An updating application does not release page locks until its verb has completed.

The reasons for holding on to the page locks until the end of the verb are fundamental to the database management system.

# 9.5 SQL Known Problems and Restrictions for Oracle Rdb Release 7.0 and Earlier

The following problems and restrictions from Oracle Rdb Release 7.0 and earlier still exist.

## 9.5.1 SQL Does Not Display Storage Map Definition After Cascading Delete of Storage Area

When you drop a storage area using the CASCADE keyword and that storage area is not the only area to which the storage map refers, the SHOW STORAGE MAP statement no longer shows the placement definition for that storage map.

The following example demonstrates this restriction:

```
SQL> SHOW STORAGE MAP DEGREES_MAP1
     DEGREES_MAP1
 For Table:             DEGREES1
 Compression is:        ENABLED
 Partitioning is:       NOT UPDATABLE
 Store clause:          STORE USING (EMPLOYEE_ID)
            IN DEG_AREA WITH LIMIT OF ('00250')
             OTHERWISE IN DEG_AREA2

SQL> DISCONNECT DEFAULT;
SQL> -- Drop the storage area, using the CASCADE keyword.
SQL> ALTER DATABASE FILENAME MF_PERSONNEL
cont> DROP STORAGE AREA DEG_AREA CASCADE;
SQL> -- Display the storage map definition.
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SHOW STORAGE MAP DEGREES_MAP1
DEGREES_MAP1 For Table: DEGREES1
Compression is: ENABLED
Partitioning is: NOT UPDATABLE
```

The other storage area, DEG_AREA2, still exists, even though the SHOW STORAGE MAP statement does not display it.

A workaround is to use the RMU Extract command with the Items=Storage_Map qualifier to see the mapping.

## 9.5.2 ARITH_EXCEPT or Incorrect Results Using LIKE IGNORE CASE

When you use LIKE...IGNORE CASE, programs linked under Oracle Rdb V4.2 and V5.1, but run under higher versions of Oracle Rdb, may result in incorrect results or %RDB−E−ARITH_EXCEPT exceptions.

To work around the problem, avoid using IGNORE CASE with LIKE or recompile and relink under a higher version (V6.0 or higher.)

### 9.5.3 Different Methods of Limiting Returned Rows from Queries

You can establish the query governor for rows returned from a query by using either the SQL SET QUERY LIMIT statement or a logical name. This note describes the differences between the two mechanisms.

If you define the RDMS$BIND_QG_REC_LIMIT logical name to a small value, the query often fails with no rows returned regardless of the value assigned to the logical. The following example demonstrates setting the limit to 10 rows and the resulting failure:

```
$ DEFINE RDMS$BIND_QG_REC_LIMIT 10
$ SQL$
SQL> ATTACH 'FILENAME MF_PERSONNEL';
SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES;
%RDB−F−EXQUOTA, Oracle Rdb runtime quota exceeded
−RDMS−E−MAXRECLIM, query governor maximum limit of rows has been reached
```

Interactive SQL must load its metadata cache for the table before it can process the SELECT statement. In this example, interactive SQL loads its metadata cache to allow it to check that the column EMPLOYEE_ID really exists for the table. The queries on the Oracle Rdb system relations RDB$RELATIONS and RDB$RELATION_FIELDS exceed the limit of rows.

Oracle Rdb does not prepare the SELECT statement, let alone execute it. Raising the limit to a number less than 100 (the cardinality of EMPLOYEES) but more than the number of columns in EMPLOYEES (that is, the number of rows to read from the RDB$RELATION_FIELDS system relation) is sufficient to read each column definition.

To see an indication of the queries executed against the system relations, define the RDMS$DEBUG_FLAGS logical name as "S" or "B".

If you set the row limit using the SQL SET QUERY statement and run the same query, it returns the number of rows specified by the SQL SET QUERY statement before failing:

```
 SQL> ATTACH 'FILENAME MF_PERSONNEL';
 SQL> SET QUERY LIMIT ROWS 10;
 SQL> SELECT EMPLOYEE_ID FROM EMPLOYEES;
 EMPLOYEE_ID
 00164
 00165
     .
     .
     .
 00173
 %RDB−E−EXQUOTA, Oracle Rdb runtime quota exceeded
 −RDMS−E−MAXRECLIM, query governor maximum limit of rows has been reached
```

The SET QUERY LIMIT specifies that only user queries be limited to 10 rows. Therefore, the queries used to load the metadata cache are not restricted in any way.

Like the SET QUERY LIMIT statement, the SQL precompiler and module processor command line qualifiers (QUERY_MAX_ROWS and SQLOPTIONS=QUERY_MAX_ROWS) only limit user queries.

Keep the differences in mind when limiting returned rows using the logical name RDMS$BIND_QG_REC_LIMIT. They may limit more queries than are obvious. This is important when using 4GL tools, the SQL precompiler, the SQL module processor, and other interfaces that read the Oracle Rdb system relations as part of query processing.

## 9.5.4 Suggestions for Optimal Use of SHARED DATA DEFINITION Clause for Parallel Index Creation

The CREATE INDEX process involves the following steps:

1. Process the metadata.
2. Lock the index name.
   Because new metadata (which includes the index name) is not written to disk until the end of the index process, Oracle Rdb must ensure index name uniqueness across the database during this time by taking a special lock on the provided index name.
3. Read the table for sorting by selected index columns and ordering.
4. Sort the key data.
5. Build the index (includes partitioning across storage areas).
6. Write new metadata to disk.

Step 6 is the point of conflict with other index definers because the system relation and indexes are locked like any other updated table.

Multiple users can create indexes on the same table by using the RESERVING table_name FOR SHARED DATA DEFINITION clause of the SET TRANSACTION statement. For optimal usage of this capability, Oracle Rdb suggests the following guidelines:

◊ You should commit the transaction immediately after the CREATE INDEX statement so that locks on the table are released. This avoids lock conflicts with other index definers and improves overall concurrency.
◊ By assigning the location of the temporary sort work files SORTWORK0, SORTWORK1, ... , SORTWORK9 to different disks for each parallel process that issues the SHARED DATA DEFINITION statement, you can increase the efficiency of sort operations. This minimizes any possible disk I/O bottlenecks and allows overlap of the SORT read/write cycle.
◊ If possible, enable global buffers and specify a buffer number large enough to hold a sufficient amount of table data. However, do not define global buffers larger than the available system physical memory. Global buffers allow sharing of database pages and thus result in disk I/O savings. That is, pages are read from disk by one of the processes and then shared by the other index definers for the same table, reducing the I/O load on the table.
◊ If global buffers are not used, ensure that enough local buffers exist to keep much of the index cached (use the RDM$BIND_BUFFERS logical name or the NUMBER OF BUFFERS IS clause in SQL to change the number of buffers).
◊ To distribute the disk I/O load, store the storage areas for the indexes on separate disk drives. Note that using the same storage area for multiple indexes results in contention during the index creation (Step 5) for SPAM pages.
◊ Consider placing the .ruj file for each parallel definer on its own disk or an infrequently used disk.
◊ Even though snapshot I/O should be minimal, consider disabling snapshots during parallel index creation.

- ◊ Refer to the Oracle Rdb7 Guide to Database Performance and Tuning to determine the appropriate working set values for each process to minimize excessive paging activity. In particular, avoid using working set parameters where the difference between WSQUOTA and WSEXTENT is large. The SORT utility uses the difference between these two values to allocate scratch virtual memory. A large difference (that is, the requested virtual memory grossly exceeds the available physical memory) may lead to excessive page faulting.
- ◊ The performance benefits of using SHARED DATA DEFINITION can best be observed when creating many indexes in parallel. The benefit is in the average elapsed time, not in CPU or I/O usage. For example, when two indexes are created in parallel using the SHARED DATA DEFINITION clause, the database must be attached twice, and the two attaches each use separate system resources.
- ◊ Using the SHARED DATA DEFINITION clause on a single−file database or for indexes defined in the RDB$SYSTEM storage area is not recommended.

The following table displays the elapsed time benefit when creating multiple indexes in parallel with the SHARED DATA DEFINITION clause. The table shows the elapsed time for ten parallel process index creations (Index1, Index2, ... Index10) and one process with ten sequential index creations (All10). In this example, global buffers are enabled and the number of buffers is 500. The longest time for a parallel index creation is Index7 with an elapsed time of 00:02:34.64, compared to creating ten indexes sequentially with an elapsed time of 00:03:26.66. The longest single parallel create index elapsed time is shorter than the elapsed time of creating all ten of the indexes serially.

***Table 9−2 Elapsed Time for Index Creations***

| Index Create Job | Elapsed Time |
|---|---|
| Index1 | 00:02:22.50 |
| Index2 | 00:01:57.94 |
| Index3 | 00:02:06.27 |
| Index4 | 00:01:34.53 |
| Index5 | 00:01:51.96 |
| Index6 | 00:01:27.57 |
| Index7 | 00:02:34.64 |
| Index8 | 00:01:40.56 |
| Index9 | 00:01:34.43 |
| Index10 | 00:01:47.44 |
| All10 | 00:03:26.66 |

## 9.5.5 Side Effect When Calling Stored Routines

When calling a stored routine, you must not use the same routine to calculate argument values by a stored function. For example, if the routine being called is also called by a stored function during the calculation of an argument value, passed arguments to the routine may be incorrect.

The following example shows a stored procedure P being called during the calculation of the arguments for another invocation of the stored procedure P:

```
SQL> create module M
cont>     language SQL
cont>
cont>     procedure P (in :a integer, in :b integer, out :c integer);
cont>     begin
cont>     set :c = :a + :b;
cont>     end;
cont>
cont>     function F () returns integer
cont>     comment is 'expect F to always return 2';
cont>     begin
cont>     declare :b integer;
cont>     call P (1, 1, :b);
cont>     trace 'returning ', :b;
cont>     return :b;
cont>     end;
cont> end module;
SQL>
SQL> set flags 'TRACE';
SQL> begin
cont> declare :cc integer;
cont> call P (2, F(), :cc);
cont> trace 'Expected 4, got ', :cc;
cont> end;
~Xt: returning 2
~Xt: Expected 4, got 3
```

The result as shown above is incorrect. The routine argument values are written to the called routine's parameter area before complex expression values are calculated. These calculations may (as in the example) overwrite previously copied data.

The workaround is to assign the argument expression (in this example calling the stored function F) to a temporary variable and pass this variable as the input for the routine. The following example shows the workaround:

```
SQL> begin
cont> declare :bb, :cc integer;
cont> set :bb = F();
cont> call P (2, :bb, :cc);
cont> trace 'Expected 4, got ', :cc;
cont> end;
~Xt: returning 2
~Xt: Expected 4, got 4
```

This problem will be corrected in a future version of Oracle Rdb.

## 9.5.6 Considerations When Using Holdable Cursors

If your applications use holdable cursors, be aware that after a COMMIT or ROLLBACK statement is executed, the result set selected by the cursor may not remain stable. That is, rows may be inserted, updated, and deleted by other users because no locks are held on the rows selected by the holdable cursor after a commit or rollback occurs. Moreover, depending on the access strategy, rows not yet fetched may change before Oracle Rdb actually fetches them.

As a result, you may see the following anomalies when using holdable cursors in a concurrent user environment:

◊ If the access strategy forces Oracle Rdb to take a data snapshot, the data read and cached may be stale by the time the cursor fetches the data.
  For example, user 1 opens a cursor and commits the transaction. User 2 deletes rows read by user 1 (this is possible because the read locks are released). It is possible for user 1 to report data now deleted and committed.
◊ If the access strategy uses indexes that allow duplicates, updates to the duplicates chain may cause rows to be skipped, or even revisited.
  Oracle Rdb keeps track of the dbkey in the duplicate chain pointing to the data that was fetched. However, the duplicates chain could be revised by the time Oracle Rdb returns to using it.

Holdable cursors are a very powerful feature for read−only or predominantly read−only environments. However, in concurrent update environments, the instability of the cursor may not be acceptable. The stability of holdable cursors for update environments will be addressed in future versions of Oracle Rdb.

You can define the logical name RDMS$BIND_HOLD_CURSOR_SNAP to the value 1 to force all hold cursors to fetch the result set into a cached data area. (The cached data area appears as a "Temporary Relation" in the optimizer strategy displayed by the SET FLAGS 'STRATEGY' statement or the RDMS$DEBUG_FLAGS "S" flag.) This logical name helps to stabilize the cursor to some degree.

| Contents